Computer Programming

Md. Masud Rana
CE 130024

Basic of Computers

RAM:

1. RAM stands for Random Access Memory.
2. It is a form of computer Data storage.
3. RAM is normally associated with volatile types of memory
4. RAM requires a flow of electricity to retain data.
5. RAM is the memory available for the operating systems, programmes and processes to hold the information during the running of the computer.
6. Writing data to a RAM chip is a faster process.

ROM:

1. ROM stands for Read Only Memory.
2. It is a form of computer data storage.
3. It is non-volatile.
4. ROM will retain data without the flow of electricity
5. Writing data to a ROM chip is a slow process.
6. Example of ROM is system BIOS.

High Level Language:

1. High Level Language is are easy to learn and modify.
2. High level Language are slow in execution.
3. High Level Language do not provide much facility at hardware level.
4. This Language are normally used to write applications programms.
5. Examples of High Level Language are: C, C++, Fortran, Java, Pascal, Pithon etc.

Low Level Language:

1. Low Level Language are difficult to learn.
2. Programmes in low level Language are fast in execution.
3. Programmes in low level Language are difficult to modify as well.
4. This Language are normally used to write Hardware programmes.

5. Example of Low level Language are:

① Machine Language, and.

⑪ Assembly Language.

⊞ Machine Language:

1. Machine code or Machine Language is a set of instruction executed directly by a computer controll processing unit (CPU).

3. Machine Language are the language understood by the computers.

3. Machine Language are almost impossible for the human to use. Because it is interly consist of numbers 0 (zero) and 1 (one).

⊞ Compiler:

A compiler is a computer programm that transforms source code written in a programm Language into another computer language.

It translates source code from a high level programming Language to a low level programming

Language.

In general, compilers are a specific types of translator.

Decompiler:

A programms that translates from a low level language to High level language is known as decompiler.

Difference between Compiler and Interpreter:

Compiler	Interpreter
Compiler ~~take~~ takes entire programms as input.	Interpreter ~~take~~ takes single Instruction as input.
Condition control statement execute faster.	Condition control statement execute slower.
Memory requirements is high.	Memory requirements is low.

Central processing unit (CPU):

The central processing unit (CPU) is the brain of the computer where most calculations takes place.

Unit of Memory:

① BIT:

② BYTE:

1 BYTE = 8 BIT
1024 BIT = 1 KILOBIT

Debugging:

Debugging is a process of finding and reducing / removing the number of bugs or deffects in a computer programm or a pitch of electric hardware.

To debug a programm or hardware deviee, one should start with the problem, isolate the problems and then fix it.

Documentation:

The program documentation is a kind of documentation that gives a compressive procedural description of a program. It shows as how a software is written.

Central processing unit (CPU):

The central processing unit (CPU) is the brain of the computer where most calculations takes place.

Unit of Memory:

① BIT:

② BYTE:

1 BYTE = 8 BIT
1024 BIT = 1 KILOBIT

Debugging:

Debugging is a process of finding and reducing / removing the number of bugs or deffects in a computer programm or a pitch of electric hardware.

To debug a programm or hardware device, one should start with the problem, isolate the problems and then fix it.

Documentation: The program documentation is a kind of documentation that gives a compressive procedural description of a program. It shows as how a software is written.

Introduction to C

History of C:

1. C is a general purpose, imperative computer programming language.
2. C was originally developed by Dennies Ritchie between 1969 to 1973.
3. It is used to implement in unixs operating system.
4. C has been standarized by the American National standard Institute (ANSI).
5. Many later languages has been borrowed directly or indirectly from C including Java, Limbo, LPC, Java Serept, PHP, Object Oriented C (OOC), C++, Pithon etc.

Advantages of C Language:

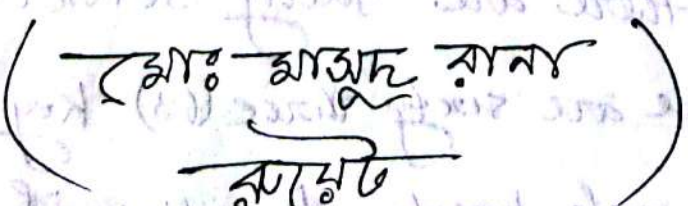
1. C is a compiler based language rather than interpreter based. So, C is a compiled in speed.
2. C is a portable language.
3. It has huge collection of library files.

4. It is easy to learn for beginners.
5. It supports good graphics.
6. It supports a number of operators.

⊞ Importance of learning C for civil Engineers:

1. Civil engineers often deals with the design, Construction and maintainance of the physically and numerically built environment. For examples: Roads, Bridge, Canals, Dams, Buildings etc.

2. It requires huge computational efforts.
3. It is almost impossible for the civil engineers to analyze such structure manually. So, it justifies the importance of learning C for civil engineers.



Constant, Variables, Data Type and Modifier

Tokens:

1. Tokens means symbol.
2. c/c++ consist of several statements.
3. Each statement again consist of one or more than one words or characters.
4. Such words and characters are jointly called token.

List of tokens in c/c++ are as follows:

1. Keywords.
2. Identifier
3. Constants
4. Strings
5. Punctuation
6. Special symbols.
7. Operator, operant and Expressions.

① Keywords: (Reserved word)

a. Keywords are reserved words.

b. In ANSI C, there are fourty seven (47) Keywords and in c++, there are sixty three (63) key words.

For C: int, char, auto, break, do, while, if, scanf, printf etc. (Follow book/Internet).

For C++: class, delete, new, private, public etc.

② Identifier:

a. Identifier refers to the name of the variables, functions, arrays, pointer, class, objects etc.

b. Variable is also an Identifier. → Justify it (Exam).

c. Example: i, j, k, a, b, c etc.

③ Constants:

a. Constants reffers to a fixed value that does not alter or change during the execution a programm.

b. Constants are two types.

(i) Integer Constants. and.

(ii) Real / Floating point constants.

① Integer Constant: It reffers to a sequence of digits. Example: +13, −11, 21, 15 etc.

Full number
No decimal.

(ii) Real floating point constant: It reffers to a sequence of digits containing decimal.

Example: −0.001, +1.235, 0.05, 6.5 etc.

④ Strings:

① Character constant can be divided into
   a. single character constant or character constant.
   b. String constant.

ⓐ Single character / character constant:

① Example: 'a', 'i', 'b', 'c', 'j' etc.
② Enclosed by single coat (' ').
③ There is no space between two like 'civilruet'.

ⓑ String constant:

① Example: "ruet", "shear", "moment" etc.
② Enclosed by double coat (" ").
③ There may exist space between two like "civil ruet".

Backslash character constant:

There are some special backslash character constants that are used in output functions.

Ex:

Backslash character constant	Meaning
\n	Indicates a new line
\b	Indicates backspace
\t	Indicates Tab / Horizontal Tab
\v	Indicates vertical Tab.

Variables:

1. Variable is the name or address of memory.
2. It is a place to store information.
3. It is important to mention data type before each variable.

Variable declaration:

The process of giving name of a variable with data type is known as variable declaration.

Formate: Datatype variablename;

Ex: int i, j, k;
float x, y; [Memory তে space নেয়]
double shear; [Memory বেশি space নেয়]

Rules for naming variable:

1. Variables must begin with a letter. (a to z)
2. Only alphabetic character (a to z), digit (0-9) and under score (_) can be used.
3. No blank space within a variable is allowed.
4. Keywords can not be used as variable.
5. Upper case and lower case are significant.
6. Length of character is normally eight (8). [Maximum 31 allowed by ANSI]

Operator, Operant, Expression:

Expression: $y = mx + c$

Operator: (=, +)

Operant: y, m, x, c

Punctuation:

It is used to differentiate keywords, identifier, operator and operant.

Example: ; : ete.

Special Symbols:

It is used for special words.

Example: #, { ?, [ ] ete.

Data Type:

1. User defined data type: structure, union, class ete
2. Built in data type: int, char, float, double ete.
3. Derived data type: array, function, pointer ete.

Assigning values to variable:

Values can be assigned using assignment operator.

Ex: x = 10;

y = x = 12;

⊞ Assignment and variable deeleration:

```
int x=10;
int x=y=12;
```

⊞ Initialization of variables:

The process of giving value to a variable is known as initialization of variables.

Example:
```
i = 0;
j = 1;
k = 2; ete.
```

⊞ Defining symboly constant:

A constant is defined as follows:

 # Format: #defined symbolicname value of constant.

Example:
```
#define pi 3.1415g
#define max 60
```

⊞ Rules for #define statement:

1. Symbolic name has the same form as the variable name.
2. No blank space is permitted between # and define.
3. # must be the first character in the line.
4. A blank space is required between #define and Symbolicname and ~~value~~ between value of constant.

5. #define statement must not end with a semicolon.
6. Symbolic name should not be assigned or used anywhere in the programme as other variables.
7. #define can be used anywhere in the programme.

Input/Output Statement

Read/ Scan data:

In c data can be read from the key board or from a file.

#Format: Scanf("List of Format specifier", &variable_list);

Example: Scanf ("%d", &a);

- → int হলে d লিখতে হয়
- → float হলে f লিখতে হয়
- → single character হলে c লিখতে হয়।
- → String হলে s লিখতে হয়।

Scanf("%d %d", &a &b);

Write/Print data:

#Format: printf("List of Format specifier", variable_list);

Ex: ⓐ Printf("%d", a);

ⓑ Printf("%d %d", p, q);

Data Files:

Necessity:

1. Large data entry from the keyboard is time consuming.
2. The entered data are lost as soon as the program is terminated.
3. Mistakes may be occured.

Opening a File:

Format: FILE *filepointer*

```
filepointer = fopen("File Name", "File Mode");
---------- = ------(---------,---------);
```

```
fclose(filepointer);
```

Example: FILE *fp;

```
fp = fopen("class test.dat", "r");
                .txt
                .doc
```

```
fclose(fp);
```

Input/output operation of files:

1. fscanf( );
2. fprintf( );

File Mode:

File Mode	Name of the Mode	Use/ Application
"w"	Write Mode	Open a file to write data
"r"	Read Mode	Open a file to read data
"a"	Append Mode	open a file to add data

fscanf( ):

To read data from a file "fscanf( )" function is used.

Format: FILE *filepointer;

fscanf(filepointer, "format specifier", &variable_list);

Example: FILE *fp;

fscanf(fp, "%d", &i);

Write("w") and Read("r") combination:

Example: FILE *fp;

fp = fopen("class test.dat", "r");

fscanf(fp, "%d", &i);

fclose(fp);

fprintf( ):

To write data to a file fprintf() function is used.

Format: FILE *filepointer;

fprintf(filepointer, "Format specifier", variable list);

Example: FILE *fp;

fprintf (fp, "%d," a);

Combined example:

```
FILE *fp;
fp= fopen("class test.dat," "w");
fprintf (fp, "%d," i);
fclose (fp);
```

Operator And Expression

operator: An operator is a logical symbol that tells the computer to perform certain mathematical or logical operations.

Types: operators are two types.

1. Unary operators.
2. Binary operators.

① Unary operators: Operator works with only one operant is known as unary operator.

Example:

x2 = +x1 → Used to indicate the positive value of any operant.

x2 = -x1 → Used to indicate the negative value of any operant.

x++ , ++x } → Add 1 to the operant.

x-- , --x } → Substruct 1 to the operant.

② Binary operator: Binary operator works with two operant.

Types: ① Arithmatical operator
② Relational operator.
③ Logical operator
④ Assignment operator
⑤ Conditional operator.
⑥ Bitwise operator

① Arithmatical operator:

Ex: + → Addition (5 = a+b)
- → substruction (5 = a-b)
* → Multiplication (5 = a*b)
/ → division (5 = a/b) → ভাগফল নির্ণয় করে।
% → Modulo division (5 = a % b) → ভাগশেষ কত হবে তা নির্ণয় করার কাজ।
( ) → Braces (5 = (a+b)(a-b)).

(ii) Relational operator:

Ex: < → Less than (x<y)

<= → Less than equal (x<=y)

> → Greater than (x>y)

>= → Greater than equal (x>=y)

== → Equal to (x==y, expression)

!= → Not equal to (x!=y)

(iii) Logical operator:

Ex: || → Logical or (x||y)

&& → Logical and (x&&y)

! → Logical not (x!y)

(iv) Assignment operator:

= → y=x+5

+= → (y+=5 or y=x+5)

-= → (y-=5 or y=x-5)

= → (y=5 or y=x*5)

/= → (y/=5) or y=x/5)

%= → (y%=5) or y=x%5)

>>= → (y>>=5 or y=x>>5)

(V) Conditional Operator:

Format: Expression-1 ? Expression-2 : Expression-3
if the condition is true Expression-2 will return else Expression-3 will return.

Ex: 12>7 ? a : b

10!=5 ? 4 : 3

(VI) Bitwise operator:

& → Bitwise and

| → Bitwise or

^ → Bitwise exclussive or

<< → Bitwise less than.

Expression

Algebric Expression	C Expression
$y = 2x+4$	y = 2 *x+4
$\frac{ax}{yz}$	(a*x)/(y*z)
$y = \sin x$	y = sin (x)
$y = \cos x$	y = cos (x)
$y = \tan x$	y = tan (x)
$y = \sin^{-1} x$	y = a sin (x)
$y = \cos^{-1} x$	y = a cos (x)
$y = \tan^{-1} x$	y = a tan (x)
$y = e^x$	y = exp (x)
$y = \lvert x \rvert$	y = fabs (x)
$y = \log x$ or $\ln x$	y = log (x) or ln (x)
$y = \log_{10}(x)$	y = log 10 (x)
$y = \sqrt{x}$	y = sqrt (x)
$y_1 = x^y$	y1 = pow (x, y)
x rounded up	ceil (x)
x rounded down	floor (x)

Controll statements

1. Conditional Controll statement.
2. Loop controll statement.

① Conditional controll statements

ⓐ if statement / simple if statement
ⓑ if... else statement.
ⓒ elseif statement / elseif Ladder
ⓓ switch statement.

ⓐ if / simple if statement:

Format:

```
if (condition)
{
  Block-01;
}
Block-2;
```

[if this condition is false then nothing to display]

[Remaining part of the program]

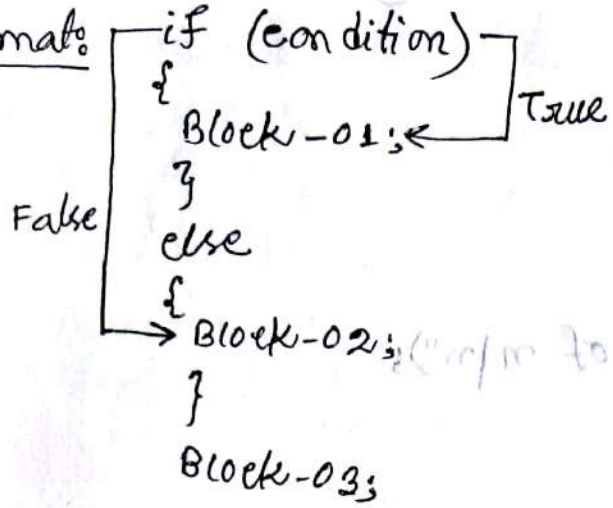
Program-01: A program to display the grade of a student.

```
#include <stdio.h>
int main()
{
  int mark;
  Printf ("Enter the mark\n");
  Scanf ("%d", &mark);
  if (mark >= 80)
  {
    Printf ("your grade is A+\n");
  }
  Printf ("Your grade is B\n");
  return (0);
}
```

if / simple if statement is used only for single condition.

⑥ if... else statement:

Format:



Program-02: A program to display the grade of a student.

```
#include <stdio.h>
int main()
{
  int mark;
  printf("Enter the mark\n");
  scanf("%d", &mark);
  if (mark >= 80)
  {
   printf("your grade is A\n");
  }
  else
  {
   printf("your grade is B\n");
  }
  return(0);
}
```

Program-03: A program to display a given full number is odd or even.

```
#include <stdio.h>
int main()
{
  int n;
  printf("Enter the value of n\n");
  scanf("%d", &n);
  if (n%2 == 0)
  {
   printf("n is Even\n");
  }
  else
  {
   printf("n is odd\n");
  }
  return(0);
}
```

Program-04: A program to count the number of students whose marks are below the average.

```
#include <stdio.h>
int main()
{
  float s1, s2, s3, s4, ave;
  int count n1, n2, n3, & n4;
  printf("Enter the marks obtained\n");
  scanf("%f%f%f%f", &s1, &s2, &s3, &s4);
  ave = (s1+ s2+s3+ s4)/ 4.0 ;
```

```
if (s1 < ave)
{
n1 = 1;
}
else
{
n1=0;
}
if (s2 < ave)
{
n2 = 1;
}
else
{
n2=0;
}
if (s3 < ave)
{
n3=1;
}
else
{
n3 =0;
}
if (s4 < ave)
{
n4=1;
}
else
{
n4=0;
}
Count = (n1+n2+n3+n4);
printf("The number of students = %d," count);
return (0);
}
```

Nested if... else statements

Format:

```
if (condition)
{
  if (condition-2)
  {
    Block-01;
  }
  else
  {
    Block-02;
  }
else
{
  Block-03;
}
```

যখন condition এর under এ একটি condition থাকবে তাকে Nested if... else statement বলে।

elseif statement:

Program-05: A program to display the grade of a student following the system applied in RUET.

```
#include <stdio.h>
int main()
{
  float mark;
  printf("Enter the mark obtained \n");
  scanf("%f", &mark);
  if (mark >= 80)
  printf("your grade is A+ \n");
  elseif (mark >=75)
  printf("your grade is A \n");
  else if (mark >= 70)
```

```
printf ("your grade is A-\n");
elseif (mark >= 65)
printf (" your grade is B+\n");
elseif ( mark >= 60)
printf (" your grade is B\n");
elseif (mark>=55)
printf (" your grade is B-\n");
elseif (mark>=50)
printf (" your grade is C+\n");
elseif (mark>=45)
printf (" your grade is C\n");
elseif (mark>=40)
printf (" your grade is D\n");
else
printf (" your grade is F\n");
return (0);
}
```

switch statement:

Format: datatype indexvalue

```
switch (switch expression)
{
case value 1:
{ Block-01;
  break;
}
case value 2:
{ Block 02;
  break;
```

```
case value 3:
{
Block-03;
break;
}
- - - - - - - - - - - -
- - - - - - - - - - - -
case value n:
{
Block n;
break;
}
default:
{
default block;
}
}
```

Program-06: A program to display the grade of a student following the system applied in RUET.

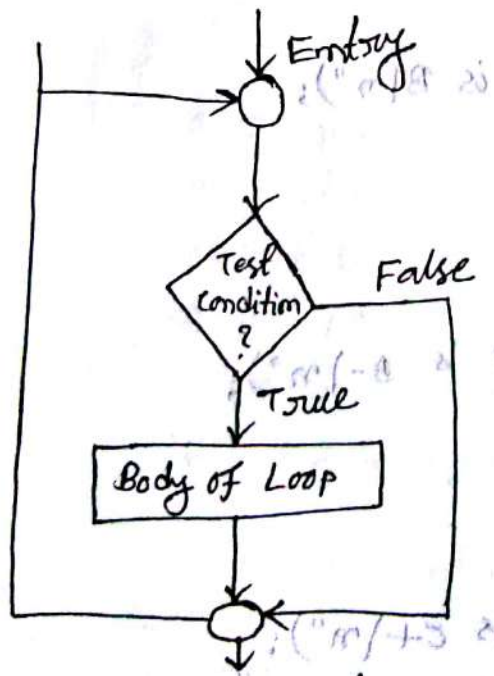
```
#include <stdio.h>
int main()
{
float mark;
int indexmax, index;
printf("Enter the mark \n");
scanf("%f", &mark);
indexmax = ~~index~~ mark/5;
if (indexmax >= 16)
{
index = 16;
}
else
{ index = indexmax;
}
```

```
switch (index)
{
case 16:
  {
   printf (" your grade is A+\n");
   break;
  }
case 15:
   {
    printf (" your grade is A\n");
    break;
   }
case 14:
    {printf (" your grade is A-\n");
     break;
    }
case 13:
     {printf (" your grade is B+\n");
      break;
     }
case 12:
     {printf (" your grade is B\n");
      break;
     }
case 11:
     {
      printf ("your grade is B-\n");
      break;
     }
case 10:
     {
     printf (" your grade is C+\n");
     break;
     }
```

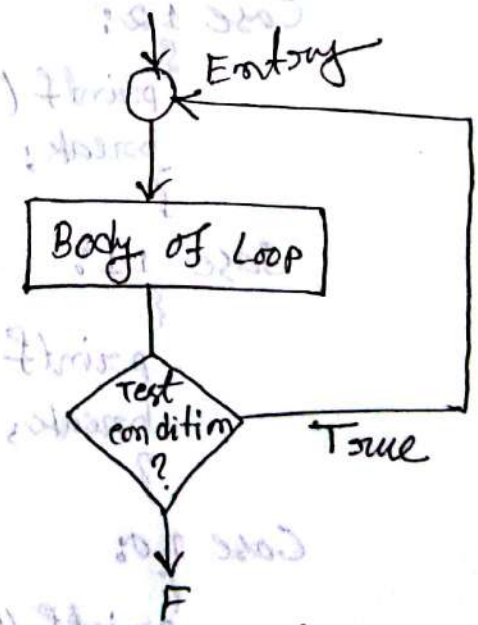
```
Case 9:
    {
    printf("your grade is C\n");
    break;
    }
Case 8:
    {
    printf("your grade is D\n");
    break;
    }
default:
    {
    Printf("your grade is F\n");
    break;
    }
  return (0);
}
```

Loop Controll statements.

1. Entry Loop controll statement.
2. Exit Loop controll statement.



Entry loop controll statement.



Exit Loop controll statement

Loop controll statement in C:

1. for statement (popular and widely used)
2. while statement.
3. do... while statement.
4. goto statement

① for statement:

Format: for(counter initialization; Test condition; counter increment/-decrement)

```
{
 Body of Loop;
}
```

Example:

```
for (i=1; i<=10; i++)
{
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
}
```

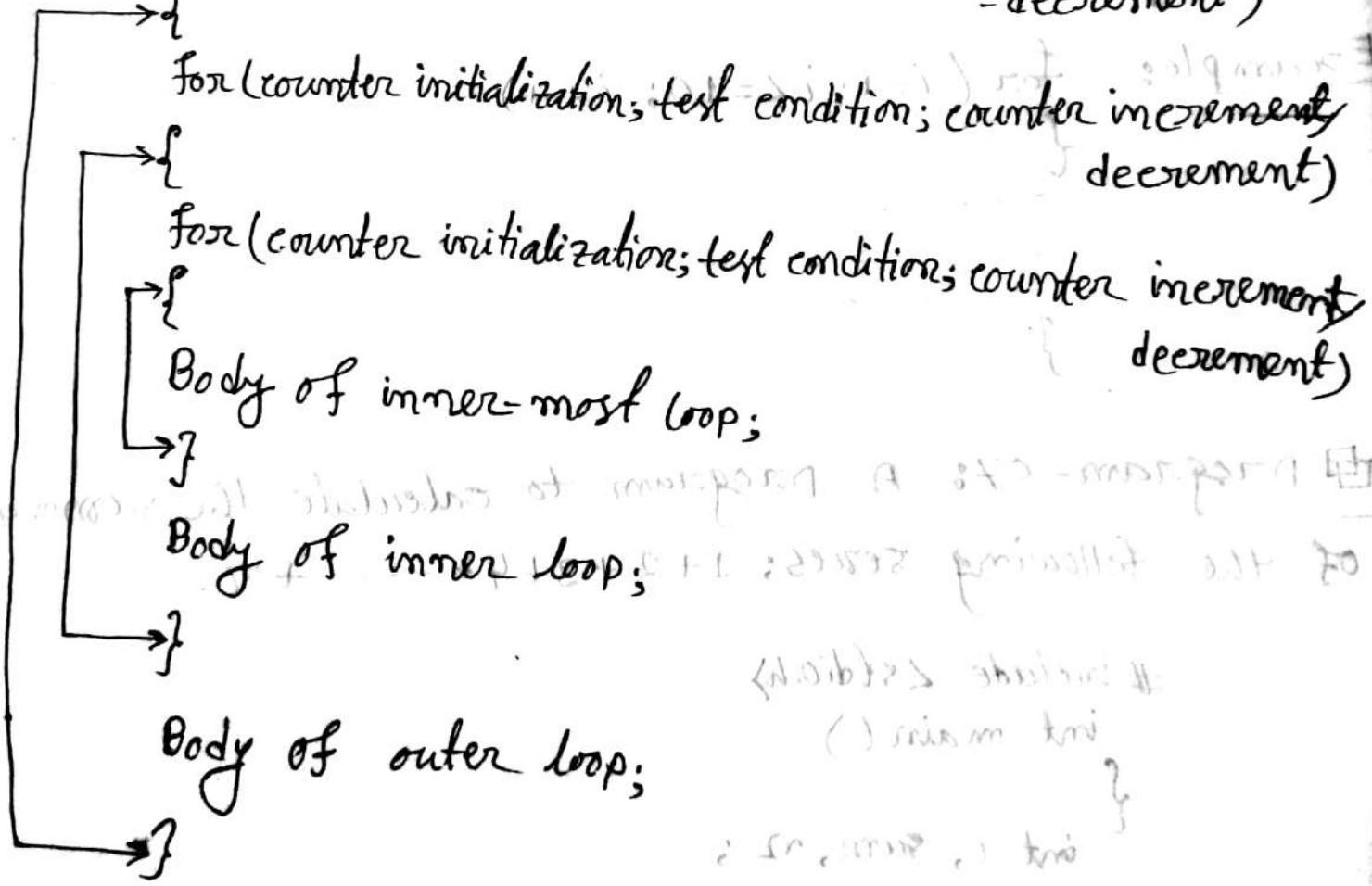

Program-07: A program to calculate the summation of the following series; 1+2+3+4+----+n.

```
#include <stdio.h>
  int main()
 {
   int i, sum, n;
   Printf("Enter the value of n\n");
   Scanf("%d", &n);
   sum=0;
   for (i=1; i<=n; i++)
```

```
{
  sum = sum + i;
[*NB: এখানে printf দিলে বার বার sum দেখাবে।]
}
printf("sum = %d", sum);
return(0);
}
```

Nested for loop:

Format: for(counter initialization; test condition; counter increment/-decrement)



Example:

```
for (i=0; i<n; i++)
{
 for (j=0; j<n; j++)
{
  for (k=0; k<n; k++)
  {
   ---------
  }
  ---------
  ---------
  }
  ---------
  ---------
  }.
```

Application of Nested for:

```
/* A program to depict the use of Nested for loop */
#include <stdio.h>
 int main()
 {
  int i,j,k,n,sum;
  printf("Enter N\n");
  scanf("%d", &n);
  sum=0;
  for (i=0; i<n; i++)
  {
   for (j=0; j<n; j++)
  {
    for (k=0; k<n; k++)
```

```
{
sum=k;
}
sum = sum+j;
}
sum = sum+i;
}
Printf("sum = %d", sum);   (1+2+3+---+N)
return(0);
}.
```

How it works?

n=3

sum=0

i=0, j=0, k=0

sum=0

i=0, j=0, k=1

sum=1

i=0, j=0, k=2

sum=2.

---

i=0, j=1, k=0

sum=0

i=0, j=1. k=1

sum=1

i=0, j=1, k=2

sum=2

sum=2+1=3

i=0, j=2, k=0

sum=0

i=0, j=2, k=1

sum=1

i=0, j=2, k=2

sum=2

sum=2+2=4

---

i=1, j=0, k=0

sum=0

i=1, j=0, k=1

sum=1

i=1, j=0, k=2

sum=2

i=1, j=1, k=0

sum=0

i=1,

⊞ while statement: (Entry controll loop statement)

Format:

```
counter initialization ;
while (test condition)
{
Body of while loop;
Counter increment / decrement ;
}.
```

Example:

```
i = 1;
while (i<=n)
{
Body ------ ;
i++;
}.
```

⊞ Application of while statement:

```
/* A program to show the application of while stat. */
#include <stdio.h>
int main ()
{
int, i, n, sum;
printf (" Enter N\n");
scanf (" %d", &n);
i = 1;
sum = 0;
while (i<=n)
{sum = sum + i;
i = i++;
}
```

```
Printf("1+2+3+---+N = %d", sum);
return (0);
}
```

Nested while statement:

Format:

```
Counter initialization;
while (test condition1)
{
while (test condition-2)
{
while (test condition-3)
{
Body of inner most loop;
counter increment/decrement;
}
Body of inner loop;
Counter increment/decrement;
}
Body of outer loop;
Counter increment/decrement;
}
```

Example:

```
i=0;
j=0;
k=0;
while (i<=n)
{
while (j<=n)
{
```

```
while (k<=n)
{
 Body ------ ;
 k++;
 }
 Body ----- ;
 j++;
 }
 Body ------ ;
 i++;
}
```

Application of Nested while statement:

```
/* A program to show the application of Nested while
      statement */

#include <stdio.h>
 int main()
 {
  int i,j,k, n, sum;
  Printf ("Enter N|n");
  Scanf ("%d", &n);
  i=0; j=0; k=0;
  sum = 0;
  {
   while (i<=n)
   {
   while (j<=n)
```

```
{
while ( k <= n )
{
  sum = sum + k;
  k = k++;
}

sum = sum + j;
j = j++;
}

sum = sum + i;
i = i++;
}
printf("1+2+3+---+n = %d", sum);
return(0);
}
```

do... while statement: (Exit controll loop statement)

Format:

```
Counter initialization;
do
{
Body of do... while statement;
Counter increment/decrement;
}
while(test condition);
```

Example:

```
i=1;
do
{
--------
--------
i=i+2;
}
while(i<=n);
```

Application of do... while statement:

/* A program to show the application of do... while statement */

```
#include<stdio.h>
int main()
{
int i, n, sum;
printf("Enter N/n");
scanf("%d", &n);
i=1;
sum=0;
```

do... while Statement: (Exit controll loop statement)

Format:

```
Counter initialization;
do
{
Body of do... while statement;
Counter increment/decrement;
}
while (test condition);
```

Example:

```
i = 1;
do
{
-------
-------
i = i + 2;
}
while (i <= n);
```

Application of do... while statement:

/* A program to show the application of do... while statement */

```
#include <stdio.h>
int main()
{
int i, n, sum;
printf("Enter N/n");
scanf("%d", &n);
i = 1;
sum = 0;
```

```
do
{
sum = sum + i;
i = i+1;
}
while (i<=n);
printf ("1+2+3+---+n = %d", sum);
return (0);
}.
```

Nested do... while statement:

Format:

goto statement:

① Format:

```
Level: (variable Name)
---------
---------
goto level;
---------
---------
```

② Format:

```
---------
---------
goto level;
---------
---------
Level:
---------
---------
```

Example:

```
---------
---------
start:
---------
---------
goto start;
---------
---------
```

Application of goto statement:

/* A program to show the application of goto statement */.

```
#include <stdio.h>
int main ()
{
  int, i, n;
  again:
  printf ("Enter N\n");
  scanf ("%d", &n);
  if (n<0)
  {
  goto again;
  }
```

```
else
{
i = i+1;
}
printf("i = %d", i);
return(0);
}
```

যদি Input ভেতর ভুল হয় তাহলে এই code এ ঠিক দিবে।

⊞ Continue statement:

Format:

```
continue;
```

Example: /* A program to find the sum of the following series; 1+2+3+5+--+n */

```
#include <stdio.h>
int main()
{
int i, n, sum=0;
printf("Enter N\n");
scanf("%d", &n);
for(i=0; i<=n, i=i+1)
  {
  if (i==4)
    {
    continue;
    }
  sum=sum+i;
  }
```

```
    printf (" sum = %d", sum );
    return (0);
}
```

Break statement:

Format:

```
break;
```

Example: See the example of switch statement.

ARRAY

Masud Rana
RUET - CE

Definition: An array is a group of data items that share a common name.
It is used to handle similar types of data.

Types of array:

1. One dimensional array.
2. Multi-dimensional array.

One dimensional array:

Format: Datatype ArrayName[Array-size];

where, Datatype indicates types of data used.

ArrayName indicates Any valid name of array variable

Array size indicates the size of array variable.

Example: int shear[5], moment[5];

shear[0], shear[1], shear[2], shear[3], shear[4].

Initialization of array variable:

① Direct initialization:

Example: int shear[5] = {10, 11, 16, 20, 15}

shear[0] = {10}

int shear[ ] = {10, 11, 16, 20, 15}

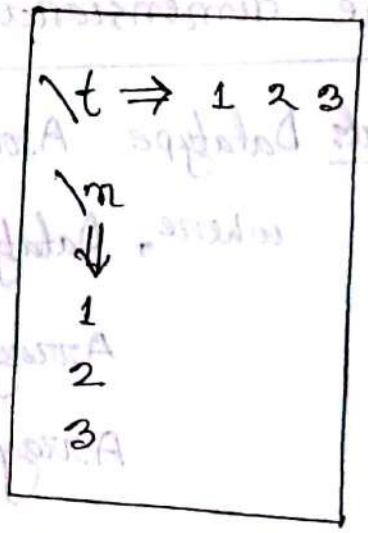
② Using for:-

```
for (i=0; i<5; i++)
  {
  scanf("%d", &a[i]);
  }
```

(i=0; a[0]
- - - - -
i=4; a[4])

Program-1:

```
/* A program to show the application of
   one dimensional array */
#include <stdio.h>
int main()
{
  int a[5], i;
  for (i=0; i<5; i++)
    {
    scanf("%d", &a[i]);
    }
// print the data
  for (i=0; i<5; i++)
    {
    printf("%d\n", a[i]);
    }
  return (0);
}
```



Program-2: /* A program to read and write one dimensional array from file */

```
#include <stdio.h>
int main()
{
  FILE *f1, *f2;
  f1 = fopen("Input.text", "r");
  f2 = fopen("Output.text", "w");
  int i, a[60];
  for(i=0; i<60; i++)
     {
      fscanf(f1, "%d", &a[i]);
     }
  for(i=0; i<60; i++)
     {
      fprintf(f2, "%d", a[i]);
     }
  fclose(f1);
  fclose(f2);
  return(0);
}
```

Multi-dimensional array:

Formate: Datatype ArrayName[ArraySize1][ArraySize2]---[ArraySizeN]

Example: int shear[3][3];

Here, Number of element = 3 x 3 = 9

Elements are:

Shear[0][0]	Shear[0][1]	Shear[0][2]
Shear[1][0]	Shear[1][1]	Shear[1][2]
Shear[2][0]	Shear[2][1]	Shear[2][2]

Initialization of two dimensional Array:

① Direct Method:

```
int shear[2][2] = {{1,2},{2,7}}
                = {
                    {1,2}
                    {2,7}

                = {1,2,2,7}.
```

② Using for/while loop:

Example:

```
for (i=0; i<3; i++)
{
    for (j=0; j<3; j++)
    {
        scanf("%d", &a[i][j]);
    }
}
```

Program-3:

```
/* A program to read and write two
   dimensional array */
#include <stdio.h>
int main()
```

```
{
  int i,j,a[3][3];
  printf("Enter the matrix\n");
  for(i=0; i<3; i++)
    {
     for(j=0; j<3; j++)
        {
         scanf("%d", &a[i][j]);
        }
    }
  printf("The matrix is:\n");
  for (i=0; i<3; i++)
    {
     for (j=0; j<3; j++)
        {
         printf ("%d\t", a[i][j]);
        }
      printf ("\n");
    }
  return (0);
}
```

Program-4: /* A program to add two matrices a&b */

```
#include <stdio.h>
  int main ()
{
  int i, j, a[3][3], b[3][3], c[3][3];
  printf("Enter the 1st matrix:\n");
```

```
for (i=0; i<3; i++)
  {
    for (j=0; j<3; j++)
      {
        scanf("%d", &a[i][j]);
      }
  }
Printf("Enter the 2nd matrix:\n");
for (i=0; i<3; i++)
  {
    for (j=0; j<3; j++)
      {
        scanf("%d", &b[i][j]);
      }
  }
for (i=0; i<3; i++)
  {
    for (j=0; j<3; j++)
      {
        c[i][j] = a[i][j] + b[i][j];
      }
  }
Printf("New matrix is:\n");
for (i=0; i<3; i++)
  {
    for (j=0; j<3; j++)
      {
        printf("%d\t", c[i][j]);
      }
    Printf("\n");
  }
return (0);
}
```

program-5: /* A program to

a) check whether matrix multiplication is possible?
b) Scan and print matrix A and B.
c) Display new matrix C obtain from multiplication of matrices A and B */

```
#include <stdio.h>
a)  int main()
{
int i, j, k, m, n, o, p, sum;
again:
printf("Enter row and column of 1st matrix\n");
scanf("%d %d", &m, &n);
printf("Enter row and column of 2nd matrix\n");
scanf("%d %d", &o, &p);
if (n!=o)
{
printf("Row of 1st matrix is not equal to the column
        of 2nd matrix\n");
Printf("Reetify error");
goto again;
}
b)  int a[m][n], b[o][p], c[m][p];
printf("Enter the 1st matrix\n");
for(i=0; i<m; i++)
  {
  for(j=0; j<n; j++)
    {
    scanf("%d", &a[i][j]);
```

```
    }
  }
Printf("Enter the 2nd matrix\n");
for(i=0; i<0; i++)
  {
   for(j=0; j<p; j++)
     {
      scanf("%d", &b[i][j]);
     }
  }
Printf("1st matrix is:\n");
for(i=0; i<m; i++)
  {
   for(j=0; j<m; j++)
     {
      Printf("%d", a[i][j]);
     }
   Printf("\n");
  }
Printf("2nd matrix is:\n");
for(i=0; i<0; i++)
  {
   for(j=0; j<p; j++)
     {
      printf("%d", b[i][j]);
     }
   printf("\n");
  }
```

c) // initialization of matrix c

```
for(i=0; i<m; i++)
  {
   for(j=0; j<p; j++)
```

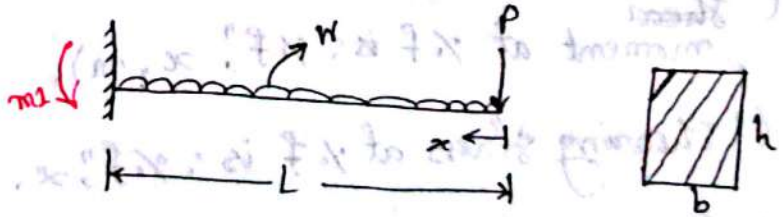
```
        {
          c[i][j] = 0;
        }
      }
// matrix multiplication
    for (i=0; i<m; i++)
      {
        for (j=0; j<p; j++)
          {
            sum=0;
            for (k=0; k<n; k++)
              {
                sum = sum + a[i][k] * b[k][j];
              }
            c[i][j] = sum;
          }
      }
    printf ("New matrix is:\n");
    for (i=0; i<m; i++)
      {
        for (j=0; j<p; j++)
          {
            printf ("%d\t", c[i][j]);
          }
        printf ("\n");
      }
    return (0);
  }
```

program-06: /* A program to calculate transpose matrix to a given matrix of any size */

```
#include <stdio.h>
 int main()
{
 int i,j,r,c;
 printf("Enter row and column\n");
 scanf("%d%d",&r,&c);
 int a[r][c], b[c][r];
 printf("Enter the matrix\n");
 for(i=0; i<r; i++)
   {
    for(j=0; j<c; j++)
      {
       scanf("%d",&a[i][j]);
      }
   }
  for(i=0; i<r; i++)
     {
      for(j=0;j<c; j++)
        {
         b[i][j]=a[j][i];
        }
      }
printf("The transpose matrix is:\n");
for(i=0; i<r; i++)
   {
    for(j=0; j<c; j++)
      {
       printf("%d\t", b[i][j]);
```

```
        }
        printf("\n");
    }
    return(0);
}

d = c+2b;
for(i=0; i<p; i++)
  {
    for(j=0; j<c; j++)
      {
        d[i][j] = c[i][j]+(2*b[i][j]);
      }
  }
  printf("%d\t", d[i][j]);
  }
  return(0);
}
```

Program-07:



① Find shear, moment, shearing stress, flexural stress at every L/10.0 distance from the free end.

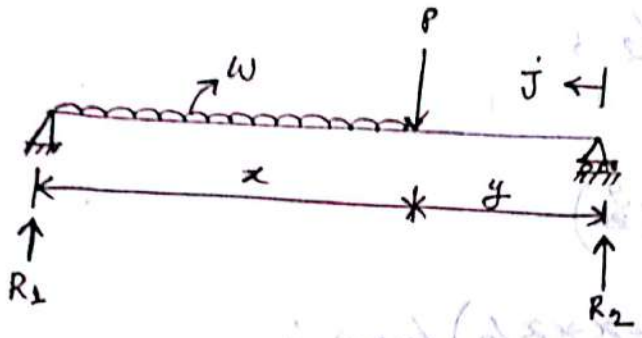
② Find maximum moment.

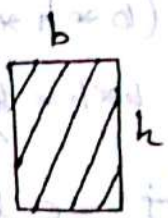
```
#include <stdio.h>
  int main ()
{
 float P,L,w,b,h, c, q, S,i, m, SS, Sf ;
 printf ("Enter the value of P,L,w, b,h \n");
 scanf ("%f %f %f %f %f", &P, &L, &w, &b, &h);
 c = h/2.0 ;
 i = b*h*h*h/12.0 ;
 q = b*h*h/8.0 ;
 for (x=0 ; x<=L ; x=x+ L/10.0)
 {
    S = P+ w*x ;
    m = P*x + w*x*x/2.0 ;
    SS = (S*q)/(i*b) ;
    Sf = (m*c)/i ;
 printf (" shear at %f is: %f", x, S);
 printf (" Shear
          moment at %f is: %f", x, m);
 printf (" Shearing stress at %f is : %f", x, SS);
 printf (" flexural stress at %f is : %f", x, Sf) ;
 }
 return (0);
}
```

// maximum moment
(Befor return(0))

```
ms= 5;
mmm = m;
m55 = 55;
m5f = 5f;
printf("maximum moment = %f", mm);
return(0);
}
```

Program-08:





① Find shear, moment, shearing stress, flexural stress at every L/10.0 distance.

② Find maximum moment.

Solution:

$$\Sigma M_1 = 0$$

$$\therefore wx \cdot \frac{x}{2} + Px - R_2(x+y) = 0$$

$$\therefore R_2 = \frac{Px + \frac{wx^2}{2}}{x+y}$$

```
#include <stdio.h>
 int main()
{
 float j, x, y, p, w, r2, r1, b, h, i, c, a, v, ss, sf, m;
 printf("Enter the value of x, y, p, w, b, h \n");
scanf("%f %f %f %f %f %f", &x, &y, &p, &w, &b, &h);
 l = x+y;
 r2 = (p*x + (w*x*x/2;)) / (x+y);
 c = h/2;
 i = (b*h*h*h)/12;
 a = b*h*h/8;
 if (j>=0 && j<y/x)
 {
    v = (p*x + w*x*x/2)/(x+y);
    m = ((p*x + w*x*x/2)/(x+y)) * j;
 }
else if (j>=0 && j<x+y)
{
   v = r2 - p - w*x;
   m = r2*j - p*j - w*x*j*j/2;
}
ss = (v*a)/(i*b);
sf = (m*c)/i;
```

```
printf("Shear at %f is: %f", j, v);
printf("moment at %f is: %f", j, m);
printf("Shearing stress at %f is: %f", j, ss);
printf("flexural stress at %f is: %f", j, sf);
return(0);
}
```

Function and program structure

Divide a large program into some small programs.

Necessity of function:

1. Every C/C++ program must have main function.
2. In case of small program, it is usual practice to write code within the main function.
3. In case of large program, it is also possible to write the whole program within the main function.
4. However, the size of main function will be so large that it is difficult to follow even for the programer.
5. Therefore, the statement of a large program can be written in more than one function.
6. It is easy to understand the code by this way.
7. It is not necessary to repeat the same portion of code and therefore it reduces the extends of error.
8. It is easy to locate and isolate the faulty location of the error.

What is function?

① When some statements are kept in a block under a name to perform certain job, it is called function.

② Every program is the collection of such one or more than one functions.

③ It is an easy way to identify the function, it is that a function ends with a pair of first braces "( )".

Example: main(), printf(), scanf() etc.

Types of function:

1. Library function.
2. User define function.

① Library function:

ⓐ It is also known as built in function.

ⓑ It should be used according to its own format.

ⓒ It is easy to use.

Example: printf, scanf etc.

② User define function:

ⓐ It is developed by the programmer according to his need.

ⓑ It depends on the type of program and the way of solving the program.

Example: sum(), ave() etc.

⊞ Elements of function:

1. Function definition.
2. Function call.
3. Function prototype. and
4. Function returntype and return statement.

⊞ ① Function definition:

ⓘ The job and the way of writting a function are described by function definition.

ⓘⓘ A user define function is composed of several statements.

ⓘⓘⓘ The definition of user define function should be located above or below the main function.

ⓘⓥ However, it is better to use it before the main function.

Format defining a user define function:-

```
returntype functionName (Argument list)
{
    Function body;
    return statement;
}
```

② Function call:

ⓐ When a function uses other function it is called main function and the function used by other is called user-define function or simply function.

ⓑ A main function can call one or more than one function or user define functions. Similarly a user define function can also call other user define function but not the main function.

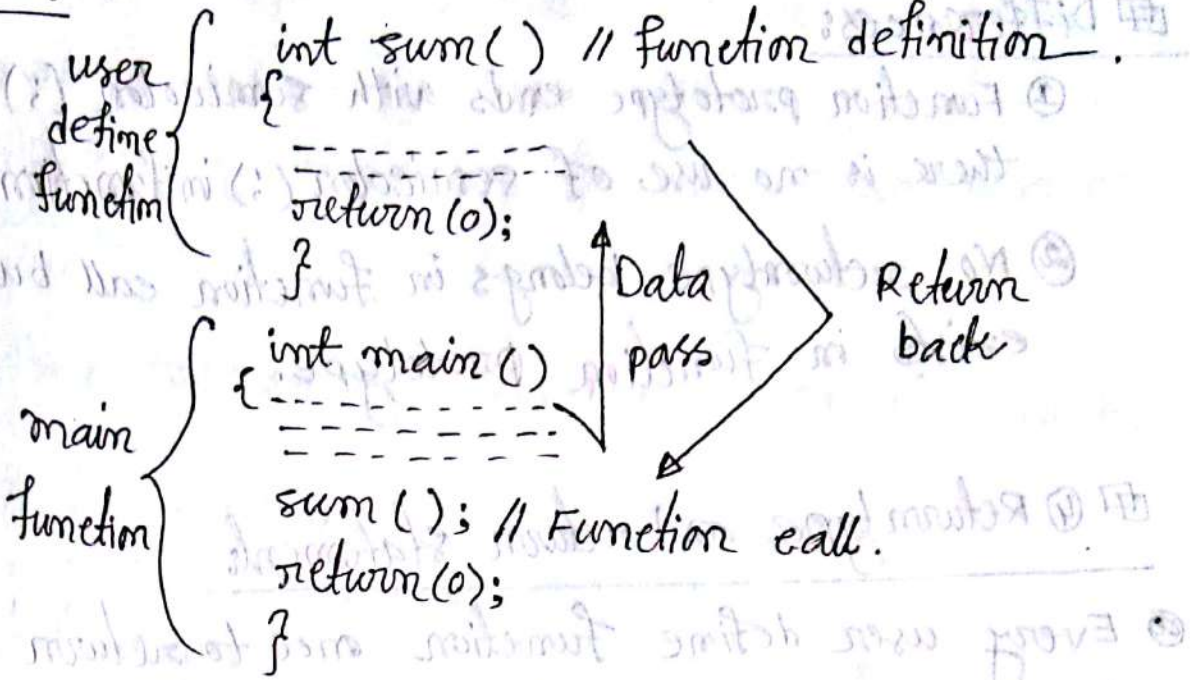
ⓒ Function call is also a statement and therefore ends with a semicolon (;).

Format of function call:

```
FunctionName( );
```

↑ Here argument list can be added if necessary.

#Example:



③ Function Prototype:

Generally function prototype is written before the main function.

#Format:

returntype functionName(argument list);

#Example:

```
int sum();
int ave(a,b);
```

returntype → functionName → argument list

Differences:

① Function prototype ends with semicolon (;) but there is no use of semicolon (;) in function definition

② No returntype belongs in function call but returntype exists in function prototype.

④ Returntype and return statement:

(a) Every user define function need to return a value to the operating system.

(b) If no returntype is declared, it is assumed that it will return int type data to the operating system.

(c) However, if void is written, it will return no data.

Format:

```
returntype functionName( )
{
- - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - -
return (return value);
}
```

N.B: No prototype is required to define before main function if user define function is written before main function. Other wise it is required.

Program: /* A program to show the use of function */.

```
#include <stdio.h>
  int sum(); // Function prototype
  int main()
{
  sum(); // Function call
  return(0);
 }
// start of user define function
  int sum()
 {
  int a, b, sum;
  printf("Enter the value of a and b\n");
  scanf("%d %d", &a, &b);
  summ = a+b;
  printf("summation = %d", summ);
  return(0);
}
```

Program: Find an inverse matrix of the given matrix ~~size~~ A[2,2] using C/C++ program.

```
/* A program to find the inverse of a matrix
   of size A[2,2] */

#include <stdio.h>
int main()
{
 int i, j;
 float determinant = 0;
 printf("Enter the matrix:\n");
 for (i=0; i<2; i++)
    {
     for (j=0; j<2; j++)
        {
        scanf("%d", &a[i][j]);
        }
    }
 printf("The matrix is:\n");
 for (i=0; i<2; i++)
    {
     for (j=0; j<2; j++)
        {
         Printf("%d", a[i][j]);
        }
        printf("\n");
    }
```

```
determinant = a[0][0] * a[1][1] - a[1][0] * a[0][1];
printf("determinat of 2x2 matrix is: %.f\n", determinant);
    int tmp = a[0][0];
    a[0][0] = a[1][1];
    a[1][1] = tmp;
    a[0][1] = 0 - a[0][1];
    a[1][0] = 0 - a[1][0];
printf ("The adjoint matrix is:\n");
    for (i=0; i<2; i++)
      {
        for (j=0; j<2; j++)
          {
            printf("%d", a[i][j]);
          }
        printf ("\n");
      }
printf ("The inverse matrix is:\n");
    for (i=0; i<2; i++)
      {
        for (j=0; j<2; j++)
          {
            printf ("%.2f\t", a[i][j] / determinant);
          }
        printf ("\n");
      }
    return (0);
}
```

Categories of function:

1. Function with no argument and no return value.
2. Function with argument but no return value.
3. Function with argument and return value.

① Function with no argument and no return value:-

Program: Write a program to find the largest value of 3 given numbers using function of category ①.

```
#include<stdio.h>
int largest();  ---> sub function
int main()
{
 largest();  ---> Function call
 return(0);
}
// starts of sub function
int largest()
{ int a, b, c, large;
 printf("Enter a, b and c\n");
 scanf("%d %d %d", &a, &b, &c);
 if(a>b && a>c)
   {
    large=a;
   }
 else if(b>c && b>a)
   {
    large = b;
   }
```

```
else
    {
     large = c;
    }
printf(" Largest = %d", large);
return (0);
}
```

② Function with argument but no return value:

program: Write a program to find the largest value of given 3 numbers using function of category ②.

```
#include <stdio.h>
  int largest (x, y, z);
  int main ()
  {
   int a, b, c;
   printf (" Enter a, b, c \n");
   scanf (" %d %d %d ", &a, &b, &c);
   largest (a, b, c);
   return (0);
  }
// start of sub function
  int largest (x, y, z)  → Dummy variable
  {
   int x, y, z, large;
   if (x>y && x>z)
```

```
    {
     large=x;
    }
  if (y>x && y>z)
    {
     large = y;
    }
else if (z>x && z>y)
    {
     large=z;
    }
Printf ("Largest = %d", large);
return(0);
}
```

③ Function with argument and return value:

Program: A program to show the use of category ③.

```
#include<stdio.h>
 int sum(intx, inty)
 main()
{
 int a, b, gsum;
 printf("Enter a and b \n");
 scanf("%d %d", &a, &b);
```

```
    gsum = sum(inta, intb) + 10;
    printf("Gross sum = %d", gsum);
    return(0);
}
int sum(intx, inty)
{
    int isum;
    isum = x+y;
    return(isum);
}
```

→ মান Transfer করা হবে main function এ। একের অধিক একই সময় Transfer করা যাবে না এজন্যে ভিন্ন ভিন্ন sub function করা হবে।

program: A program to find the largest and smallest value of three given numbers using two functions or sub functions or user define functions.

```
#include<stdio.h>
int largest(intx, inty, intz);
int smallest(intx, inty, intz);
main()
{
    int a, b, c, large, small;
    printf("Enter a, b and c\n");
    scanf("%d %d %d", &a, &b, &c);
```

```
large = largest (int a, int b, int c);
small = smallest (int a, int b, int c);
printf ("Largest value = %d", large);
printf ("smallest value = %d," small);
return (0);
}
int largest (int x, int y, int z)
{
  int ilarge;
  if (x > y && x > z)
    {
    ilarge = x;
    }
  elseif ( y > z && y > x)
    {
    ilarge = y;
    }
  else
    {
    ilarge = z;
    }
  return (ilarge);
}
  int smallest (int x, int y, int z)
  {
    int ismall;
```

```
if (x<y && x<z)
  {
   ismall = x;
  }
else if (y<x && y<z)
   {
    ismall = y;
   }
 else
   {
    ismall = z;
   }
 return (ismall);
}
```

Passing arrays as function argument in C:

There are three methods available.

⇒ Method-1:

Formate: Functiontype FunctionName(Datatype *parameters)

pointer

```
{
---------
---------
---------
}
```

Example:

```
int average(int *pL)
{
---------
}
```

⇒ Method-2:

Formate: Functiontype FunctionName(Datatype ArrayName[A.Size])

```
{
---- ------
---- ------
---- ------
}
```

Example:

```
int average(int a[10])
{
---- ------
---- ------
}
```

⇒ Method-3:

Formate: Functiontype FunctionName(Datatype ArrayName[ ])

```
{
---- ------
---- ------
---- ------
}
```

Example:

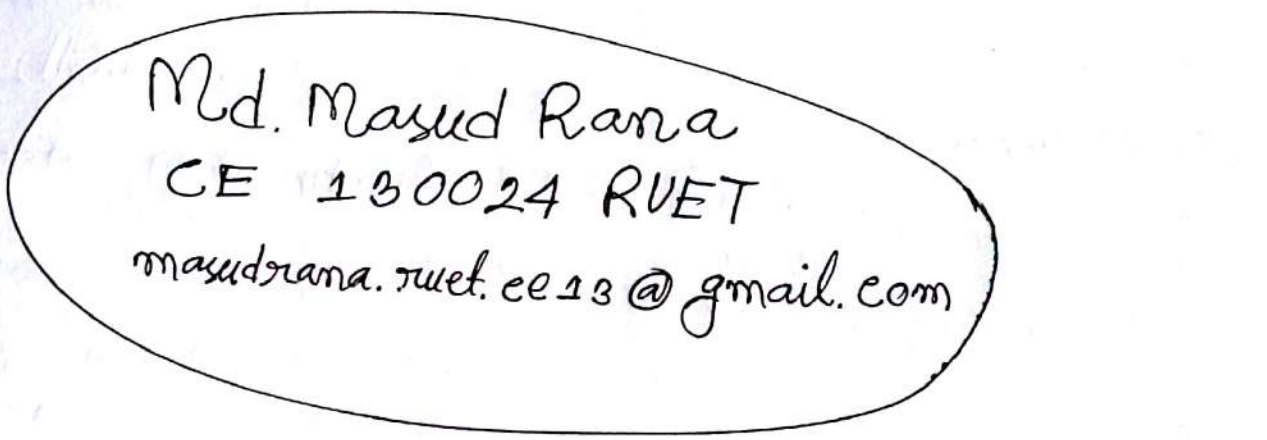
```
int average(int a[ ])
{
---- ------
---- ------
}
```

Program: Write a program to find the average of n given numbers by using function.

```
#include<stdio.h>
float average(int n, int a[10]);
int main()
{
  int i, n, a[10];
  float avg;
  printf("Enter N\n");
  scanf("%d", &n);
  for (i=0; i<n; i++)
  {
    scanf("%d", &a[i]);
  }
  avg = average(a, n)          → Function call
  printf("Average=%f", avg);
  return(0);
}

float average(int a[10], int n)
{
  int i, sum=0.0; avg;
```

(a → a[10] বলা লাগে না শুধু নাম দিয়ে call করতে হবে।)

```
for (i=0; i<n; i++)
   {
   sum = sum + a[i];
   }
iavg = sum/n ;
return (iavg);
}
```



Md. Masud Rana
CE 130024 RUET
masudrana.ruet.ce13@gmail.com

Pointer

Definition:

1. Pointer points to the location in memory.
2. A pointer is a veriable whose value is the address of another variable.
3. Pointer variable or simply ~~variable~~ pointer is the special types of variable that holds memory address rather than the data.
4. Pointers are the powerful feature of C and C++ language which prefers it from other popular languages such as java and visual basic.
5. Pointers are used in c program to access the memory and ~~multiplution~~ manipulation the address.

Reference operator and Pointer variable:

a. Reference operator are used for defining the pointer variable.

b. It is denoted by *.

⊞ Format:

Datatype Reference Operator Pointer variable name.

int *P, *test;

Program: /* A program to demonstrate the use of Referenee Operator */

```
#include<stdio.h>
int main()
{
  int a=10;
  printf("value of variable A=%d\n", a);
  printf("Address of variable A=%d\n", &a);
  return(0);
}
```

&a → Address of variable in memory.

=====: THE END :=====

PROGRAMMING - 1

1 1 2 1 4 1 6 1 . . . : N where N = 0 : 2 : 10

Solve:
```
#include<stdio.h>
void main()
{
int i,sum;
sum=0;
for(i=1;i<=20;i++)
{
if(i==3 && i==5)
continue;
sum=sum+i;
}
}
printf("%d",sum);
}
```

Output:
210

1 + 2 + 3 + 4 + . . . + N

Solve:
```
#include<stdio.h>
void main()
{
int i,sum,N;
printf("enter the value:\n");
scanf("%d",&N);
sum=0;
for(i=1;i<=N;i++)
{
sum=sum+i;
}
printf("%d",sum);
}
```

Output: N=20

enter the value:
20
210

Point
100/10

3. Write a program to find the factorial of N

Solve:

```
# include <stdio.h>
Void main ()
{
  int i, S, N;
  S= 1;
  printf("enter N= \n");
  Scanf("%d", &N);
  for (i=1 ; i<=N; i++)
  { fact=fact*i;
  }
  printf("fact = %d", S);
}
```

Output:

```
enter N= 4
4
fact = 24
```

~ o ~

RONY 100070

4. 1!+2!+3!+ · · · N!

Solve:

```
# include<stdio.h>
Void main()
{
  int i, S, N, Sum;
  sum= 0;
  fact= 1;
  printf("enter N: \n");
  Scanf(" %d", &N);
  for (i=1 ; i<=N ; i++)
  { S=S*i;
    Sum = Sum+S;
  }
  printf("Sum= %d", Sum);
}
```

Output:

```
enter N=
7
Sum=5913
```

~ o ~

** যদি প্র. থাকে যে, 1-2+3-4+ · · N তখন Simply (1+3+5+ · · · 2N-1)-(2+4+ · · · 2N) এভাবে ভাগ করে Sum1 ও Sum2 বের করে Sum= Sum1-Sum2 করতে হবে। ... Series টি এভাবে Solve করতে হবে।

5. 1! + 3! + 5! + 7! + ... (2N-1)!

Solve:
```
#include<stdio.h>
void main()
{
 int i,j,fact,N,sum;
 fact=1
 sum=0;
 printf("enter N=\n");
 scanf("%d",&N);
 for(i=1;i<=N;i=i+2)
 {
  fact=1; fact=fact*i;
  for(j=1;j<=i;j++)
  {
   fact=fact*j;
  }
  sum=sum+fact;
 }
 printf("sum=%d",sum);
}
```

Output:
enter N=
11
sum=10284847

~o~

6. 2+4!+6!+... (2i)!

Solve:
```
#include<stdio.h>
void main()
{
 int i,S,j,N,sum;
 fact=1
 sum=0;
 printf("enter N=\n");
 scanf("%d",&N);
 for(i=2;i<=N;i=i+2)
 {
  S=1;
  for(j=1;j<=i;j++)
  {
   S=S*j;
  }
  sum=sum+S;
 }
 printf("sum=%d",sum);
}
```

Output:
enter N=
8
sum=41066

RONY 100070

~o~

7. $1^2 + 2^2 + 3^2 + \cdots + N^2$

Solve:
```
#include<stdio.h>
void main()
{
int i, N, sum;
sum=0;
printf("enter N=");
scanf("%d",&N);
for(i=1; i<=N; i++)
{ sum = sum + pow(i,2);
}
printf("sum = %d", sum);
}
```

Output: enter N=
4
sum = 30

~o~

8. $1^2 + 3^2 + 5^2 + \cdots + (2N-1)^2$ (i=i+2)

Solve:
```
#include<stdio.h>
void main()
{
int i, N, sum;
sum=0;
printf("enter N=");
scanf("%d",&N);
for(i=1; i<=N; i=i+2)
{ sum = sum + pow(i,2);
}
printf("Sum = %d", sum);
}
```

Output: enter N=
5
sum = 35

RONY 100070

~o~

9. $2^2 + 4^2 + 6^2 + \cdots + (2N)^2$ ⟶ Same to 8 only (i=i+2)

10. $\frac{1}{1^2} + \frac{2}{2^2} + \frac{3}{3^2} + \cdots + \frac{N}{N^2}$

Solve:
```
#include<stdio.h>
void main()
{
float i, N, sum;
sum=0.0;
printf("enter N=\n");
scanf("%f", &N);
for(i=1; i<=N; i++)
{ sum=sum+(i/pow(i,2));
}
printf("sum=%f", sum);}
```

Output: enter=
3
Sum = 1.83333

~o~

Write a program that reads a positive integer $N \leq 50$ and calculate $\frac{1}{1^2} + \frac{3}{2^2} + \frac{5}{3^2} + \frac{7}{5^2} + \frac{11}{6^2} + \cdots \frac{2N-1}{N^2}$ 2005

Solve:

```
#include <stdio.h>
# include <math.h>
# include <conio.h>
void main()
{
float i,N,Sum;
Sum= 0.0 ;
printf ("enter N = \n");
scanf ("%f",&N);
if (N >50)
printf ("invalid input");
else
{
for (i=1; i<=N; i++)
{ if (i==4)
continue;
Sum = Sum +(2i-1)/pow(i,2);
}
printf ("Sum=%f", sum);
}
}
```

Output:

```
enter N=
15
Sum=1.618518
```

RONY 100070

$\frac{2^2}{3^3} + \frac{4^2}{5^3} + \frac{6^2}{7^3} + \cdots + \frac{(2N)^2}{(2N+1)^3}$

Solve:

```
#include <stdio.h>
void main()
{
float i,N,Sum;
Sum= 0.0;
printf("enter N = \n");
scanf("%d",&N);
for (i= 2; i<= N; i=i+2)
{ Sum = Sum +(pow(i,2)/pow((i+1),3)); }
printf ("Sum = %f", Sum); }
```

Output:

```
enter N=
10
Sum = 0.541027
```

13) $\frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{N!}$

Solve:

```
#include <stdio.h>
void main()
{
float i,S,sum,N;
sum = 0.0;
S = 1.0;
printf("enter N:= \n");
scanf("%f",&N);
for(i=1,i<=N; i++)
{
S = S*i
sum = sum + (1/S);}
printf("sum = %f", sum);
}
```

Output:

```
enter N=
5
Sum = 1.716667
```

14) $\frac{1}{1!} + \frac{1}{3!} + \frac{1}{5!} + \cdots + \frac{1}{(2N-1)!}$

Solve:

```
#include <stdio.h>
Void main()
{
float i,j,N,S,sum;
sum=0.0;
printf("enter=\n");
scanf("%f",&N);
for(i=1; i<=N; i=i+2)
{
S=1.0;
for(j=1; j<=i; j++)
{ S = S*j;
}
sum = sum+(1.0/S);
}
printf("Sum= %f", sum);
}
```

Output:

```
enter=
3
Sum= 1.16667
```

RONY 100070

15. $\frac{1}{1!} + \frac{1}{3!} + \frac{1}{5!} + \frac{1}{7!} + \cdots + \frac{1}{27!}$

Solve:

```
#include <stdio.h>
void main()
{
float i,j,s,sum;
sum=0.0;
for(i=1 ; i<= 27; i=i+2)
{
  s=1.0 ;
  for(j=1. ; j<=i; j++)
 { s=s*j;
 }
 sum= sum + (1.0/s);
}
printf("sum = %f", sum);
 }
```

Output:

sum = 1.175201

RONY 100070

16. $\frac{1}{1!} + \frac{3}{3!} + \frac{5}{5!} + \cdots + \frac{(2N-1)}{(2N-1)!}$

Solve:

```
#include <stdio.h>
void main()
{ float i,j,s,N,sum;
printf("enter N=\n");
scanf("%f",&N);
sum= 0.0;
for(i=1;i<=N;i=i+2)
{ s=1.0;
 for(j=1;j<=i;j++)
 { s=s*j;
 }
 sum = sum + (i/s);
 }
printf("sum= %f", sum);
 }
```

Output:

```
enter N=
5
Sum= 1.541667
```

17. $\frac{1}{2!} + \frac{1}{4!} + \frac{1}{6!} + \cdots + \frac{1}{30!}$ 2007

Solve:
```
#include <stdio.h>
void main()
{
  float i,j,S, sum;
  sum=0.0;
  for (i=2; i<=30; i=i+2)
  { S=1.0;
    for (j=1; j<=i; j++)
    { S=S*j;
    }
    sum=sum+(1/S)
  }
  printf("sum= %f",sum);
}
```

Output: sum= 1.175201

~o~

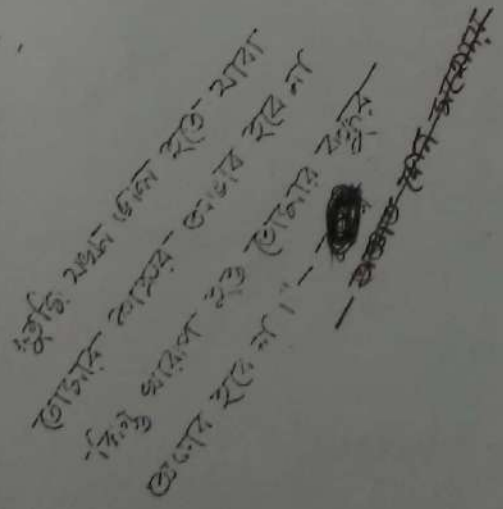
18. $\frac{1}{1!} + \frac{3}{3!} + \frac{5}{5!} + \cdots$ upto 100 numbers 2006

Solve:
```
#include <stdio.h>
void main()
{ float i,j,S,sum;
  sum=0.0;
  for(i=1; i<=100; i=i+2)
{
  S=1.0;
  for (j=1; j<=i; j++)
  { S=S*j;
  }
  sum=sum+(i/S);
  }
printf("sum = %f",sum);
}
```

Output:

sum= 1.543081

~o~



$\frac{1^N}{1!} + \frac{2^N}{2!} + \frac{3^N}{3!} + \cdots + \frac{N^N}{N!}$ 2011

Solve:

```
#include<stdio.h>
void main()
{
float i,j,s,N,sum;
printf("enter N=\n");
scanf("%f",&N);
sum=0.0;
for(i=1,i<=N;i++)
{
for(j=0;j<=i;j++)
s=s*j;
}
sum=sum+(pow(i,2)/s);
}
printf("sum=%f",sum);
}
```

Output:

```
enter N=
30
sum= 5.436564
```

RONY 100070

Write a program to find the summation of even and odd number from first ten positive integer values.

Solve:

```
#include<stdio.h>
void main()
{
int i,odd=0,even=0,n=100;
for(i=1;i<=n;i++)
{if(i%2==0)
{even=even+i;
}
else
{odd=odd+i;
}}
printf("Sum of odd=%d",odd);
printf("Sum of even=%d",even);
}
```

Output:

```
odd= 2500
even= 2550
```

2. Write a program to count the number and sum of all integers greater than 50 and less than 250 that are devided by 7. 2006, 2009

Solve:

```
#include<stdio.h>
void main()
{
  int i,s,sum=0;
  Sum=0;
  printf("number:\n");
for(i=50;i<=250;i++)
 {
 if(i%7==0)
  {
  sum=sum+i;
  }
}
```

```
for(i=56;i<=250;i+=7)
{ s
  ;
  sum=sum+i;
}
printf("Sum=%d",sum);
printf("\n number=%d",c);
}
```

Output:

number:
Sum = 4214
number = 28

3. Write a program to calculate the average of n numbers and then compute the deviation. 2004, 2008

Solve:

```
#include<stdio.h>
void main()
{
  int n,i,a[1000];
  float mean,dev,sum=0;
printf("enter n:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
  printf("enter a[i]:");
  scanf("%d",&a[i]);
sum=sum+a[i];
}
mean=sum/n;
printf("mean=%f",mean);
for(i=1;i<=n;i++)
 { dev=mean-a[i];
  printf("\ndev %d=%f",i,dev);
 }}
```

Output:

```
enter n : 5
enter a[i]:11
enter a[i]:18
enter a[i]:12
enter a[i]:16
enter a[i]:17
```

mean = 16.200001
dev 1 = 5.200001
dev 2 = -1.79999
dev 3 = -2.79999
dev 4 = 0.200001
dev 5 = -0.799999

[এই program এর মাধ্যমে 1000 টি Number নিয়ে কাজ করতে পারবে। ; A[1000] এর মধ্যে value change করা যায়। কিন্তু array ছাড়া যে কোন সংখ্যক Number নিয়ে কাজ করা যায় না।]

~ 9 ~

Write a program to calculate the average of n numbers and the calculate the numbers above and below average.

Solve:

```
#include<stdio.h>
void main()
{
  int n,i,sum,c1[1000];
  float c1=0, c2=0, ave;
  printf("enter n:");
  scanf("%d",&n);
  sum=0;
  for(i=1;i<=n;i++)
  { printf("enter c1[%d]:\n");
    scanf("%d",&c1[i]);
    sum=sum+c1[i];
  }
  ave=sum/n;
  printf("ave=%f",ave);
  for(i=1;i<=n;i++)
  { if(c1[i]>=ave)
    { c1++;
    }
    else
    { c2++;
    }}
  printf("\nabove=%f",c1);
  printf("\nbelow=%f",c2);
}
```

Output:

```
enter n: 4
enter ct[i]:
14
enter ct[i]:
13
enter ct[i]:
16
enter ct[i]:
17
ave:- 15.0000
above = 2.0000
below = 2.0000
```

30.1.2002

3.1 A class of 60 student takes an examination on CE 205 in which scores range from 0 to 100. Write a c++ program to find-

(i) highest score (ii) the number of student who failed i.e. score below 40

Solve:

```
#include <stdio.h>
#include <conio.h>
#define b 60
void main()
{
int a[b], K, r, f;
printf("Enter the marks one by one :\n");
for (K=0; K<b; K++)
{ scanf("%d", &a[K]);
}
r = a[0];
for(K=0; K<b; K++)
{ if (r<a[K])
  r= a[K];
}
printf("Highest score : %d", r);
f=0;
for (K=0; K<b; K++)
{ if (a[K]<40)
  f=f+1;
}
printf("\n No of student failed: %d", f)
}
```

Tito
RONY
100870

Write a program to find largest value from a number of integer value: 2003, 2005 (n=3)

Solve:

```
# include <stdio.h>
#define b n
void main()
{
 int b[n], K, n = a[b], K, n
printf ("Enter the values : \n");
for (K=0; K<b; K++)
{
 scanf ("%d", &a[K]);
}
n = a[0];
for (K=0; K<b; K++)
 if (n<a[K])
  n = a[K];
}
printf("Highest value : %d\n", n);
}
```

put n = যে কোন সংখ্যা

2007, 2002011 (b=120), 2001

3. Four different class tests are given to class of CE 205 of 60 students. Write a program to calculate the average of best of three class test for every student.

Solve:

```
# include<stdio.h>
#define b 60
Void main()
{
 int a[b], i;
 float ave, C11, C12, C13, C14;
 for (i=1; i<=b; i++)
```

RONY
190070

```
{ printf("\nenter c-t marks of roll %d:", a[i][j]);

printf("\n c-t 1:");
scanf("%f", &c-t 1);
printf("%f", &c-t 2);
printf("\n c-t 2:");
scanf("%f", &c-t 2);
printf("\n c-t 3:");
scanf("%f", &c-t 3);
printf("\n c-t 4:");
scanf("%f", &c-t 4);

if (c-t 1 <= c-t 2 && c-t 1 <= c-t 3 && c-t 1 <= c-t 4);
ave = (c-t 2 + c-t 3 + c-t 4)/3;
if (c-t 2 <= c-t 1 && c-t 2 <= c-t 3 && c-t 2 <= c-t 4);
ave = (c-t 1 + c-t 3 + c-t 4)/3;
if (c-t 3 <= c-t 1 && c-t 3 <= c-t 2 && c-t 3 <= c-t 4);
ave = (c-t 1 + c-t 2 + c-t 4)/3;
if (c-t 4 <= c-t 1 && c-t 4 <= c-t 2 && c-t 4 <= c-t 3);
ave = (c-t 1 + c-t 2 + c-t 3)/3;
printf("ave c-t mark of roll %d: %f", i, ave);
}}
```

RONY 1000:10

~ 9 ~

27. 15 numbers are given: i) find largest value ii) find smallest value iii) find no of odd number iv) find no of even

Solve:

```
#include<stdio.h>
#define b 15
void main()
{
int a[b], i, odd=0, even=0, largest, smallest;
for (i=0; i<b; i++)
{ printf("enter value %d=", a[b]);
  scanf("%d", &a[i]);}
  largest = a[0];
```

```
for (i=0; i<b; i++)
{ if (largest < a[i])
  largest = a[i]; }
smallest = a[0];
for (i=0; i<b; i++)
{ if (smallest > a[i])
  smallest = a[i]; }
for (i=0; i<b; i++)
{ if (a[i]%2 == 0)
{ even++;
}
else
{ odd++; }
printf ("\n largest number: %d", largest);
printf ("\n smallest number: %d", smallest);
printf ("\n even number : %d", even);
printf ("\n odd number: %d", odd); }
```

~ 9 ~

28. Using Do loop print the numbers less than 100 which are divisible by 6. : 2000

Solve: `#include <stdio.h>`

```
#define n 100
void main()
{
  int i;
  i=0;
 do
 {
  i%6==0;
  i=i+6;
  printf ("\n value= %d", i);
  }
 while (i<n-6); }
```

Output:

value = 6
value = 12
value = 18
value = 24
value = 30
value = 36
value = 42
value = 48
value = 54
value = 60
value = 66
:
value = 96

RONI 100070

~ 10 ~

2011

22/ Write a program to print the input number in a reve... order.

Solve:

```
#include<stdio.h>
void main()
{
 int  a,d1,d2,d3,d4;
printf("enter the value = ");
scanf("%d",&a);
d1=a/100;
d2= a%100;
d3= d2/10;
d4= d2%10;
printf("\n ans: %d %d %d", d4,d3,d1);
}
```

d4 d3 d1

output:

```
enter the value = 234
ans : 4 3 2
```

RONI 100%70

23/ Write a program to find the largest value out of three number without using array or using if else

Solve:

```
#include<stdio.h>
void main()
{
 int  a,b,c,d,e,f,g;
printf(" enter the value: a,b,c");
scanf("%d%d%d",&a,&b,&c);
if(a>b)
{ d=a;
 }
else
 { d=b;}
 e=d;
if(a>c)
{ f=a;}
else
{ f=c;}
g=f;
printf("\n largest value: %d",g);
}
```

output:

```
enter the value: a,b,c
2 3
3 1
5 1
largest value: 51
```

31. A hollow cylindrical soil sample having inner dia, $r_1 = 3$ cm, outer dia, $r_2 = 5$ cm, height 6 cm. Write a program to calculate hollow volume and solid volume of the soil mass.

Solve:

```
#include<stdio.h>
#define r1 3
#define r2 5
#define h 6
#define pi 3.141b
void main()
{
  float v1, v2, v3;
  v1 = pi * pow(r1,2) * h/1;
  v2 = pi * pow(r2,2) * h/1;
  v3 = pi * v2 - v1;
  printf("hollow volume = %f", v1);
  printf("\nsolid volume = %f", v3);
```

Output: hollow volume = 42.411598
solid volume = 75.398399

~ o ~

32. A program to identify a leap year or not leap year

Solve:

```
#include<stdio.h>
void main()
{
int a;
printf("enter :");
scanf("%d", &a);
if(a%4==0)
{
printf("%d is a leap year", a);
}
else
{printf("%d is not a leap year", a);}
```

Output: enter: 2000
2000 is a leap year.

RONY 130070

```
if(a%4==0 && a%400==0)
    leap year
elseif(a%100!=0)
    leap year
else
    not leap year
```

~ o ~

38. A program to calculate current bill

Solve:
```
#include<stdio.h>
void main()
{
float a,b,c,d,s;
printf("enter a,b,c\n");
scanf("%f%f%f",&a,&b,&c);
d=(a+b+c)*150/1000;
printf("d=%f KWH",d);
if(d>200)
s=(d-200)*7+(200*4);
if(d<=200)
s=(d-100)*6+(100*5);
if(d<=100)
s=100*5;
printf("\ns=%f TK",s);
}
```

output:

enter a,b,c:
100
700
400
d=150.00 KWH
s=980.00 TK

~o~

Ark
22.02.13

MD LUTFOR RAHMAN
RONY
100070

CT-10 (No. of stdnt get Highest mark)

39. A program to calculate avg of ct marks of n number of students and find the no of students who obtain below or above avg number.

Solve:
```
#include<stdio.h>
void main()
{int n,i,sum,ct[100],c1=0,c2=0;
float ave;
printf("enter n:");
scanf("%d",&n);
sum=0;
for(i=1;i<=n;i++)
{printf("enter ct mark of roll %d:",i);
scanf("%d",&ct[i]);
sum=sum+ct[i];
}
ave=sum/n;
printf("\navg=%f",ave);
for(i=1;i<=n;i++);
{if(ct[i]>=ave)
c1++;
else
c2++;}
printf("\nabove=%d",c1);
printf("\nbelow=%d",c2);}
```

~o~

KNOW THYSELF

PROGRAMMING-2

MD. MUKTOR RAHMAN
ROLL
100070

2010; 2006, 2000

1. Write a program to read a matrix A[4x4] and print the same.

Solve:

```
#include <stdio.h>
void main()
{
    int i,j, A[4][4];
    printf("enter your matrix:\n");
    for(i=0; i<4; i++)
    { for(j=0; j<4; j++)
        scanf("%d",&A[i][j]);
    }
    printf("you have entered:\n");
    for(i=0; i<4; i++)
    { for(j=0; j<4; j++)
        { printf("%d\t", A[i][j]);
        }
        printf("\n");
    }
```

Output:

```
enter your matrix:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
you have entered:
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 16
```

2005(a[3x3] and b[3x3]), 2001(a[3x2] & b[3x2])

2. Write a program to add two matrices a[2x2] and

Solve:

```
#include<stdio.h>
void main()
{ int i,j,k,a[2][2],b[2][2],c[2][2],sum=0;
printf("enter 1st matrix :\n");
for( i=0 ; i<2; i++)
{ for(j=0;j<2;j++)
  scanf("%d",&a[i][j]);}
printf("enter 2nd matrix: \n");
for(i=0; i<2; i++)
{ for(j=0;j<2;j++)
  scanf("%d",&b[i][j]);}
for(i=0;i<2;i++)
{ for(j=0;j<2;j++)
  {
  for(k=1;k<2;k++)
  c[i][j]= a[i][j]+b[i][j];}
  }
printf("output matrix:\n");
for(i=0;i<2;i++)
{ for(j=0;j<2;j++)
  printf("%d\t", c[i][j]);
  printf("\n");}}
```

Output:

```
enter 1st matrix:
1
2
3
1
enter 2nd matrix:
1
2
3
1
output matrix:
2  4
6  8
```

tab

RONY
100070

3. Write a program to multiply two matrix A(3x3) and B(3x3)

Solve:

```
#include<stdio.h>
void main()
{ int i,j,k,a[3][3],b[3][3],c[3][3],sum=0;
```

```
printf("enter 1st matrix:\n");
for(i=0;i<3;i++)
{ for(j=0;j<3;j++)
  scanf("%d",&a[i][j]);}
printf("enter 2nd matrix:\n");
for(i=0;i<3;i++)
{ for(j=0;j<3;j++)
  scanf("%d",&b[i][j];}
for(i=0;i<3;i++)
{ for(j=0;j<3;j++)
  { c[i][j]=0
  for(k=0;k<3;k++)
  c[i][j]=c[i][j]+
  a[i][k]*b[k][j];}
  }}
```

```
printf("output matrix:\n");
for(i=0;i<3;i++)
{ for(j=0;j<3;j++)
printf("%d\t",c[i][j]);
printf("\n");}}
```

Output:

enter 1st matrix:

1
2
3
4
5
6
7
8
9

enter 2nd matrix

1
2
3
1
5
6
7
8
9

output matrix:

30	36	42
66	81	96
102	126	150

~ o ~

2011

4. Two matrices A[3,3], B[3,3] are given. Write a program to find a new program matrix c following the equation C = A + AB

Solve:

```
#include<stdio.h>
void main()
{
int i,j,k,a[3][3],b[3][3],c[3][3],d[3][3],e[3][3];
printf("enter 1st matrix:\n");
```

```
for (i=0; i<3; i++)
{ for (j=0; j<3; j++)
  scanf ("%d", &a[i][j]); }
printf ("enter 2nd matrix: \n");
for(i=0; i<3; i++)
{ for (j=0; j<3; j++)
  scanf ("%d", &b[i][j]); }
for (i=0, i<3; i++)
{ for (j=0; j<3; j++)
  { c[i][j]= 0
  for (k=0; k<3; k++)
  { c[i][j]= c[i][j] + a[i][k] * b[k][j]; }
  c[i][j]=0; }
for (i=0; i<3; i++)
{ for (j=0; j<3; j++)
  d[i][j]= a[i][j]; }
for(i=0, i<3; i++)
for(j=0; j<3; j++)
d[i][j]= A[i][j]+B[i][j];
printf ("output matrix: \n");
for(i=0; i<3; i++)
{ for (j=0; j<3; j++)
  printf ("%d\t", d[i][j]);
  printf ("\n"); } }
```

Output

enter 1st matrix:
1
2
3
4
5
6
7
8
9

enter 2nd matrix:
1
2
3
4
5
6
7
8
9

output matrix

31	38	45
70	86	102
109	134	159

Write a program using c program to find the matrix C(3,3) such that C = A+2AB, where A(3,3), B(3,3).

Solve:

```
#include <stdio.h>
void main()
{ int i,j,k,a[3][3],b[3][3],c[3][3],d[3][3],e[3][3];
printf ("enter 1st matrix: \n");
```

```
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
scanf("%d",&a[i][j]);}
printf("enter 2nd matrix:\n");
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
scanf("%d",&b[i][j]);}
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
{c[i][j]=0
for(k=0;k<3;k++)
c[i][j]=c[i][j]+a[i][k]*b[k][j];}
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
d[i][j]=a[i][j]+c[i][j];}
printf("output matrix:\n");
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
printf("%d\t",c[i][j]);
printf("\n");}
```

output:

enter 1st matrix:	enter 2nd matrix
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

output matrix:

61	74	87
136	167	198
211	260	309

2008

6) Three matrices A[3,3], B[3,3] and C[3,3] are given. Write a program to find a new matrix D following the equation

D = AB + 2C

Solve:

```
#include <stdio.h>
void main()
{
int i,j,k,a[3][3],b[3][3],c[3][3],d[3][3],e[3][3];
printf("enter 1st matrix:\n");
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
scanf("%d",&a[i][j]);}
printf("enter 2nd matrix\n");
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
scanf("%d",&b[i][j]);}
printf("enter 3rd matrix:\n");
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
scanf("%d",&c[i][j]);}
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
{e[i][j]=0;
for(k=0;k<3;k++)
e[i][j]=e[i][j]+a[i][k]*b[k][j];}}
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
d[i][j]=e[i][j]+c[i][j];}
printf("output matrix:\n");
for(i=0;i<3;i++)
{for(j=0;j<3;j++)
printf("%d\t",d[i][j]);
printf("\n");}}
```

RONY
1000

2007

7. Two matrices A[4,4], B[4,4] are given. Write a program to find a new matrix C following the equation C=3A+AB

Solve: Same → 5 only in all for loop use 4 instead of 3

2006

8. Three matrices A[3,3], B[3,2] and C[3,2] are given. Write a program to find a new matrix D following the equation

D = AB+C

Solve: Same as 6

2004, 2003 (A(4,3) & B(3,2))

Write a program to multiply two matrix A(4,3) and B(3,4)

Solve:

```
#include<stdio.h>
void main()
{
  int i,j,k,a[4][3],b[3][4],c[4][4],s=0;
  printf("enter 1st matrix:\n");
  for(i=0;i<4;i++)
  { for(j=0;j<3;j++)
    scanf("%d",&a[i][j]);}
  printf("enter 2nd matrix:\n");
  for(i=0;i<3;i++)
  { for(j=0;j<4;j++)
    scanf("%d",&b[i][j]);}
  for(i=0;i<4;i++)
  {for(j=0;j<4;j++)
   {s=0;
   for(k=0;k<3;k++)
   {
     s=s+a[i][k]*b[k][j];}
    c[i][j]=s;}
  printf("output matrix:\n");
  for(i=0;i<4;i++)
  { for(j=0;j<4;j++)
  printf("%d\t",c[i][j]);
  printf("\n\n\n\n");}}
```

RONI
100070

enter 1st matrix:
1 2 3 1 5 6 7 8 9 0 1 2

enter 2nd matrix:
3 4 5 6 7 8 9 0 1 2 3 4

output matrix:
20 26 32 53
53 68 [illegible] [illegible]
[illegible] 110 121 9
9 12 15 8

09.03.13

MD.LUTFOR RAHMAN

Find average and number of student DONY below average by array method:-

```
#include
void main()
{
int i, sum=0, n=0, average;
float a[100]
printf("enter the marks : \n");
for(i=0; i<100; i++)
{
scanf("%f", &a[i]);
sum=sum+a[i];
}
average=sum/10;
for (i=0; i<1000; i++)
{
if(a[i]<average)
n=n+1;
printf("number of student = %d", n);
}
}
```

$i/i+i$

$i+=2$

$i<\frac{i}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}$

$\backslash n /$

KNOW THYSELF

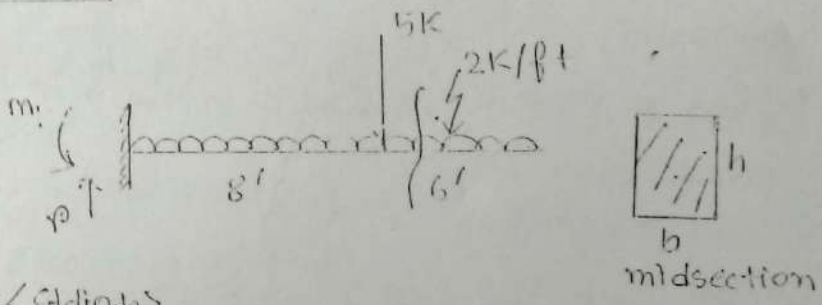
CWI

Programming-3

MD LUTFOR RAHMAN
RONY
100070

2010

1. Write a program to calculate shear force, bending moment, Shearing stress, flexural stress at an interval of 2ft from the fixed support of the catilever beam shown in fig. below:



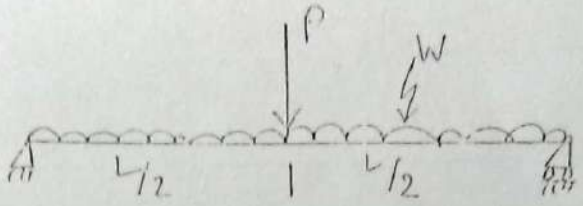
Solve:

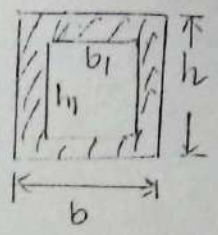
```
#include<stdio.h>
void main()
{ float e,i,b,h,v,q,m,s,f,p,m1,x;
 printf("enter \nb=");
 scanf("%f",&b);
 printf("e\nh=");
 scanf("%f",&h);
 m1 = 236;
 p = 33;
 for(x=0; x<=14; x+=2)
 { if(x>=0 && x<6)
   v = p-(2*x);
   m = -m1+(2*x*x/2);
  else
  { v= p-(2*x)-5;
    m = -m1+(2*x*x/2)+5*(x-3);
  }
 e=h/2;
 q=b*h*h;
 i=b*h*h*h/12;
 s=v*q/(i*b);
 f=m*e/i;
printf("\nm at distance %f in shear force
   =%f\n bending moment=%f\n
   Shearing stress=%f\n flexural
   stress=%f", x,v,m,s,f);}
```

RONY
100070

2011

Write a program to find (i) Shear (ii) moment (iii) Shearing stre (iv) flexural stress at every L/10 distance of the beam shown in figure below.



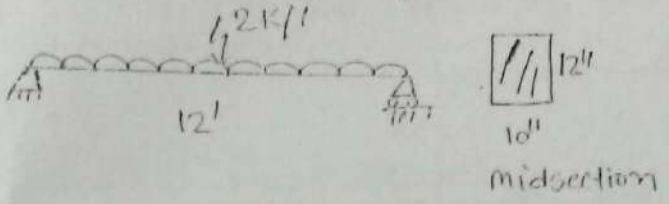


Solve:

```
#include<stdio.h>
void main()
{ float r1,r2,P,W,l,x,b,b1,h,h1,v,q,m,c,i,s,B,f;
  printf("enter l=");
  scanf("%f",&l);
  printf("\nP=");
  scanf("%f",&P);
  printf("\nW=");
  scanf("%f",&W);
  printf("\nb=");
  scanf("%f",&b);
  printf("\nb1=");
  scanf("%f",&b1);
  printf("\n h1=");
  scanf("%f",&h1);
  r1=((P*l/2)+(w*l*l/2))/l;
  r2= P+(w*l)-r1;
  c= h/2;
  q=((b*h*h/8)-(b1*h1*h1/8));
  B= b-b1;
  i= ((b*h*h*h)-(b1*h1*h1*h1))/12;
  for(x=0;x<=l;x=x+(l/10))
  { if(x==0||x<l/2)
    { v= r1-(w*x);
      m=r1*x-(w*x*x/2);}
    else
    { v=r1-P-(w*x);
      m=r1*x-P*(x-(l/2))-(w*x*x/2);}
    s=v*c/(i*B);
    f=m*c/i;
  printf("\n\n\n at distance %f\n shear=%f\n moment=%f\n s.stress=%f\n f.stress=%f",
  x,v,m,s,f);}}
```

ROLL NO
100710

Write a program to calculate the shearing stress and bending stress using the following figure:



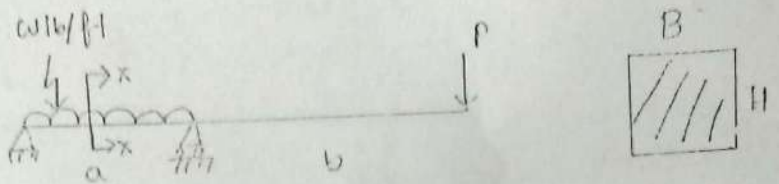
Soln:

```
#include<stdio.h>
#define w 2
#define l 12
#define b 0.833
#define h 1.00
void main()
{ float x,r1,r2,v,q,i,c,m,f,s;
  printf("enter:\nx=");
  scanf("%f",&x);
  r1=(w*l*l/2)/l;
  r2=w*l-r1;
  c=h/2;
  i=b*h*h*h/12;
  q=b*h*h/8;
  if(x<0&&x>l)
  printf("invalid input");
  else
  { v=r1-w*x;
    m=r1*x-(w*x*x/2);
    s=v*q/(i*b);
    f=m*c/i;
  printf("\nat distance %f\nSS=%f\n BS=%f",s,f);}
```

RONY
100070

2008, 2005

Write a general program to calculate shear force, bending, shearing stress and flexural stress at any where of the follow beam.

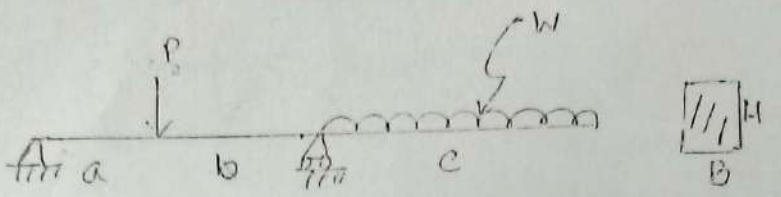


Solve:

```
#include<stdio.h>
void main()
{ float s,f,r1,r2,v,q,b,i,m,c,x,w,a,B,H,P;
printf("enter: \nx=");
scanf("%f",&x);
printf("\na=");
scanf("%f",&a);
printf("\nb=");
scanf("%f",&b);
printf("\nw=");
scanf("%f",&w);
printf("\nP=");
scanf("%f",&P);
printf("\nB=");
scanf("%f",&B);
printf("\nH=");
scanf("%f",&H);
r2=(w*a*a/2+P*(a+b))/n;
r1= P+w*a-r2;
q= B*H*H/8;
c= H/2;
i= B*H*H*H/12;
if(x<0 && x>(a+b))
printf("invalid input");
else
if(x>=0 && x<a)
{ v= r1-w*x;
m= r1*x - w*x*x/2;}
else if(x>=0 && x<=(a+b))
{ v=r1+r2-w*(x-a);
m= r1*x -w*(x-a)*(x-a)/2 + r2*(x-a);}
s= v*q/(i*B);
f= m*c/i;}
printf("\n at distance %f\nshear=%f\nbmoment=%f
\nbr stress: %f\nbm stress=%f",x,v,m,s,f);}
```

RONY
100070

Write a general program to calculate shear force, bending moment, shearing stress and flexural stress of the following beam.



Solve:

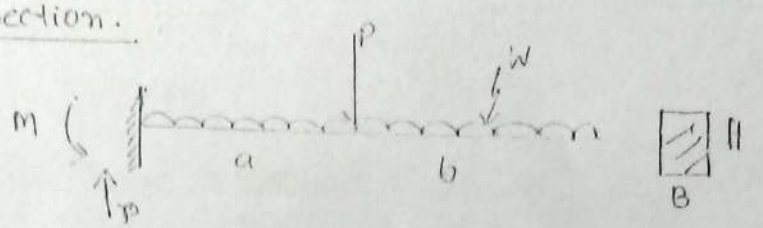
```
#include<stdio.h>
void main()
{float p,a,b,c,w,B,H,v,q,x,i,m,s,f,C,r1,r2;
printf("enter: \n p=");
scanf("%f",&p);
printf("\n a=");
scanf("%f",&a);
printf("\n b=");
scanf("%f",&b);
printf("\n c=");
scanf("%f",&c);
printf("%f\n w=");
scanf("%f",&w);
printf("\n B=");
scanf("%f",&B);
printf("\n H=");
scanf("%f",&H);
printf("\n x=");
scanf("%f",&x);
r2=(p*a+w*c*(b+c/2))/(a+b);
r1= p+w*c-r2;
C= H/2;
i= B*H*H*H/12;
q= B*H*H1B;
if(x<0||x>(a+b+c))
printf("invalid input");
else
{if(x>=0&&x<a)
{ v=r1;
  m=r1*x;}
else if(x>=a&&x<(a+b))
{ v= r1-p;
  m=r1*x - p*(x-a);}
else if(x>=(a+b)&&x<=(a+b+c))
{ v= r1-p+r2 - w*c;
  m=r1*x - p*(x-a)+r2*(x-a-b) - w*(x-a-b)*
                                (x-a-b)/2;}
  s= v*q/(i*B);
  f= m*C/i;
printf("\n at distance %f\n Shear=%f\n B.moment=%f\n
S.Stress=%f\n f.Stress=%f",x,v,m,s,f);}}
```

RONY
100070

Scanned by CamScanner

2006

Write a program to calculate the shear, moment, shear stress and flexural stress of a beam as shown in fig. below at any section.



Solve:

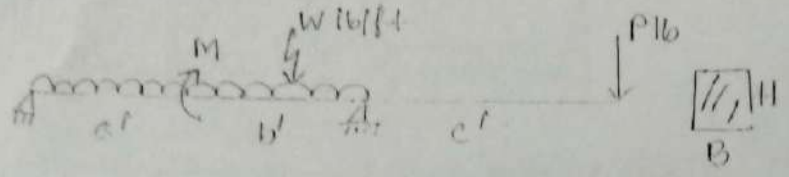
```
#include<stdio.h>
void main()
{ float a,b,P,W,B,H,c,RP,V,m,q,S,f,x,I;
  printf("enter: \na=");
  scanf("%f",&a);
  printf("\nb=");
  scanf("%f",&b);
  printf("\nP=");
  scanf("%f",&P);
  printf("\nW=");
  scanf("%f",&W);
  printf("\nB=");
  scanf("%f",&B);
  printf("\nH=");
  scanf("%f",&H);
  printf("\nx=");
  scanf("%f",&x);
  RP = P + W*(a+b)
  m = -M + RP*x - P*(x-a) - W*x*x/2;
  c = H/2;
  q = B*H*H/8;
  I = B*H*H*H/12;
```

```
if(x<0 && x>(a+b))
  printf("invaid input");
else
{ if(x>=0 && x<a)
  { V = RP - W*x;
    m = -M + RP*x - W*x*x/2; }
  else if(x>=a && x<=(a+b))
  { V = RP - W*x - P;
    m = -M + RP*x - P*(x-a) - W*x*x/2; }
  S = V*q/(I*B);
  f = m*c/I;
Printf("\nDistance %f\nShear=%f\n
  Moment=%f\nShearing stress=%f\n
  flexural stress=%f", x, V, m, S, f); } }
```

RONY
100070

P

Write a general program to calculate shear force, bending moment, shearing stress and flexural stress at any where of the following beam.



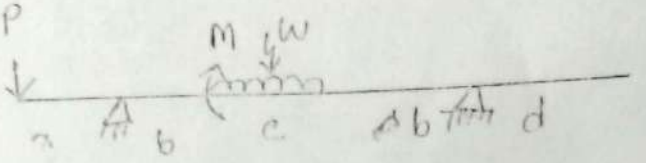
Solve:

```
#include<stdio.h>
void main()
{ float a,b,c,M,w,P,B,H,x,V,m,q,i,C,r1,r2;
  printf("enter :\na=");   C = H/2;
  scanf("%f",&a);          q = B*H*H/8;
  printf("\nb=");          i = B*H*H*H/12;
  scanf("%f",&b);
  printf("\nc=");          if(x<0 && x>(a+b+c))
  scanf("%f",&c);          printf("invalid input");
  printf("\nM=");          else
  scanf("%f",&M);          { if(x>=0 && x<a)
  printf("\nw=");            { V = r1 - w*x;
  scanf("%f",&w);            m = -r1*x + w*x*x/2; }
  printf("\nP=");          else if(x>=a && x<(a+b))
  scanf("%f",&P);            { V = r1 - w*x + w*(x-a-b) + r2;
  printf("\nB=");              m = -r1*x - M + w*x*x/2 - w*(x-a-b)*(x-a-b)/2 - r2*(x-a-b); }
  scanf("%f",&B);
  printf("\nH=");          s = V*q/(i*B);
  scanf("%f",&H);          f = m*c/i;
  printf("\n x=");        printf("\n At distance %f m Shear=%f\n B.M=%f\n
  scanf("%f",&x);                 S.F=%f\n F.S=%f", x,V,m,s,f);
r2=(-M - w*(a+b)*(b+w)/2 - P*(a+b+c))/(a+b);
r1 = w*(a+b) + P - r2;
```

RONY
100070

2003

* Write a program to calculate SF and BM anywhere the following beam.



Solve:

```
#include<stdio.h>
void main()
{ float P,a,b,M,w,c,d,x,V,m,r1,r2;
  printf("enter:\nP=");
  scanf("%f",&P);
  printf("\na=");
  scanf("%f",&a);
  printf("\nb=");
  printf("\n");
  scanf("%f",&b);
  printf("\nc=");
  scanf("%f",&c);
  printf("\nd=");
  scanf("%f",&d);
  printf("\nx=");
  scanf("%f",&x);
  printf("\nM=");
  scanf("%f",&M);
r2=(-P*a+M-0+w*c*(b+c/2))/(2b+c);
r1= P*+ w*c - r2;
if(x<0 && x>(a+b+c+b+d))
printf("Invalid input");
else
{
  if(x>=0 && x<a)
  { V=-P;
    m=--P*x;}
  else if (x>=a && x<(a+b))
  { V=-P+r1;
    m=-P*x+r1*(x-a);}
  else if (x>=(a+b) && x<(a+b+c))
  {V=-P+r1-w*(x-a-b);
   m=-P*x+r1*(x-a)+M-w*(x-a-b)*
     (x-a-b)/2;}
  else if (x>=(a+b+c+b) && x<=(a+2b+c+d))
  { V=-P+r1-w*(x-a-b)+r2;
    m=-P*x+r1*(x-a)+M-w*(x-a-b)
      *(x-a-b-c/2)+w*(x-a-b-c)*
      (x-a-b-c)/2+r2*(x-a-2b-c);}
printf("\nat m at distance %f-m
  SF=%f\n BM=%f",V,m);}}
```

RONY
100070

r for r P+0

else if (x >= (a+b+c) && x < (a+2b+c))

{ v = -P + r1 - w*a(x-a-b) + w*(x-a-b-c); - w*c

m = -P*x + r1(x-a) - w*(x-a-b)*(x-a-b)/2 + M + w*(x-a-b-c)*(x-a-b-c)/2;}

else if (x >= (a+2b+c) && x <= (a+2b+c+d))

{ v = -P + r1 - w*(x-a-b) + w*(x-a-b-c-d) + r2;

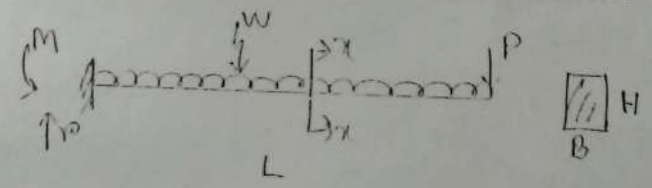
m = -P*x + r1*(x-a) + M - w*(x-a-b)*(x-a-b)/2 + w*(x-a-b-c)*(x-a-b-c)/2 + r2*(x-a-b-c-b);}

printf("\n ad distance %f\n SF=%f\n BM=%f", x, v, m);}}

~ o ~

2002

Q. Write a program to calculate SF, BM, ES, FS at an interval of 5ft from the fixed support of the following cantilever beam:



RONY
100070

Solve:

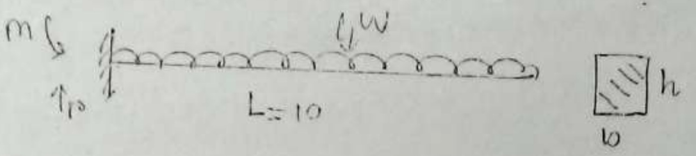
```
#include<stdio.h>
#define x 5
void main()
{
 float w,L,x=5, P, B,H ,M,r1, V,m , S,f,q,i,c
 printf("enter: \n w= ");
 Scanf("%f",&w);
 printf("\n L=");
 scanf("%f",&L);
 printf("\n P=");
 Scanf("%f",&P);
 printf("\n B=");
 scanf("%f",&B);
 printf("\n H=");
 Scanf("%f",&H);

 r1 = w*L + P;
 M = (w*L*L/2 + P*L);
 c = H/2;
 i = B*H*H*H/12;
 q = B*H*H/8;
 if(x<0 && x>L)
 Printf("invalid input");
 else
 { V = r1 - w*x;
   m = -M + r1*x - w*x*x/2;}

 for(x=0; x<l; x=x+5)
 {
 V = r1 - w*x;
 m = r1*x - w*x*x/2;
 }
 Ss = (V*Q)/(i*b);
 Sf = (m*c)/i;
```

```
s = v*q/(I*b);
f = m*c/i;
printf("\n at distance %f\n SF = %f\n BM=%f\n SF= %f\n FS= %f", x,v,m,
);
```

~o~

2017.

Write a program to calculate the maximum SS and BS at every 0.2 m distance interval.



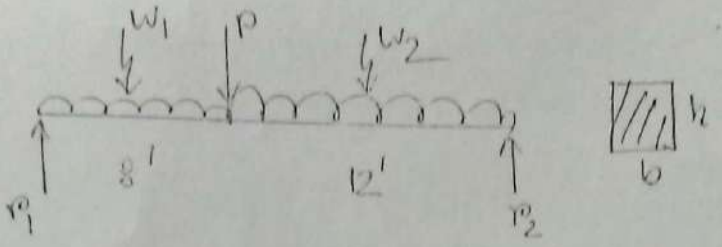

```
#include<stdio.h>
#define L 10
void main()
{ float w, b, h, M, R, s, f, v, q, i, m, x, c;
  printf("enter:\n w=");
  scanf("%f", &w);
  printf("\n b=");
  scanf("%f", &b);
  printf("\n x=");
  scanf("%f", &x);
  printf("\n h=");
  scanf("%f", &h);
  R = w*L;
  M = w*L*L/2;
  for(x=0; x<=L; x+=0.2)
  { v= R - w*x;
    m= R*x - M - w*x*x/2;
    c = h/2;
    i = b*h*h*h/12;
    q = b*h*h/8;
    s = v*q/(i*b);
    f = m*c/i;
  printf("\n at distance %f \n SS=%f
          \n BS = %f", x, s, f); }}
```

~o~

RONI
100070

1.4

Calculate BM, SS, FS, SF at any point using CP



Solve:

```
#include<stdio.h>
#define a 8
#define b 12
void main()
{ float r1,r2,w1,w2,P,bh,c,V,m,S,f,q,i,x ;
printf("enter:\nw1=");
scanf("%f",&w1);
printf("\nw2=");
scanf("%f",&w2);
printf("\nP=");
scanf("%f",&P);
printf("\nb=");
printf-scanf("%f",&b);
printf("\nh=");
scanf("%f",&h);
r1 = (w1*a*(b+a/2)+P*b+w2*b*b/2)/(a+b);
r2 = w1*a+w2*b+P-r2;
c = h*r2;
q = b*h*h/2;
i = b*h*h*h/12;
if(x<0&&x<(a+b))
printf("invalid input");
```

RONY
420070

```
else
{ if (x >= 0 && x < a)
  { v = v0 - w1*x;  m = v0*x - w1*x*x/2; }
else if (x >= a && x < (a+b))
  { v = v0 - w1*x + w1*(x-a) - P - w2*(x-a);
    m = v0*x - w1*x*x/2 + w1*(x-a)*(x-a)/2 - P*(x-a) - w2*(x-a)*(x-a)/2
    ;}
  s = v*q/(i*b);
  f = m*c/i;
printf("\nmat distance %f %f\n SF=%f\n BM=%f\n SS=%f\n BS=%f\n %f", x, m, s, f);}
```

Asif

18-03-13

MD. LUTFOR RAHMAN

RONY

100070