Output Problems

1. What is the output of this program?

```
#include<stdio.h>
int main()
{
        int i = 0;
        while (i<= 4)
        {
        printf("%d", i);
        if (i> 3)
                gotoinside_foo;
        i++;
        }
        return 0;
}

void foo()
{
inside_foo:
        printf("PP");
}
```

Output: Compiler error: Label “inside_foo” used but not defined.
Explanation: Scope of a label is within a function. We cannot goto a label from other function.

2. What is the output of this program?

```
#include<stdio.h>
#define a 10
int main()
{
#define a 50
printf("%d",a);

getchar();
return 0;
}
```

Output: 50
Preprocessor doesn’t give any error if we redefine a preprocessor directive. It may give warning though. Preprocessor takes the most recent value before use of and put it in place of a.

3. What is the output of this program?

```
int main()
{
```

```
        char str[] = "geeksforgeeks";
        char *s1 = str, *s2 = str;
        int i;

        for(i = 0; i< 7; i++)
        {
                printf(" %c ", *str);
                ++s1;
        }

        for(i = 0; i< 6; i++)
        {
                printf(" %c ", *s2);
                ++s2;
        }

        getchar();
        return 0;
}
```

Output :
g ggggggg e e k s f
Explanation:
Both s1 and s2 are initialized to str. In first loop str is being printed and s1 is being incremented, so first loop will print only g. In second loop s2 is incremented and s2 is printed so second loop will print “g e e k s f “

4.What is the output of this program?

```
#include<stdio.h>
int main()
{
        char str[] = "geeksforgeeks";
        int i;
        for(i=0; str[i]; i++)
                printf("\n%c%c%c%c", str[i], *(str+i), *(i+str), i[str]);

        getchar();
        return 0;
}
```

Output:
gggg
eeee
eeee

kkkk
ssss
ffff
oooo
rrrr
gggg
eeee
eeee
kkkk
ssss
Explanation:
Following are different ways of indexing both array and string.
arr[i]
*(arr + i)
*(i + arr)
i[arr]
So all of them print same character.

5.What is the output of this program?

```
#include<stdio.h>
int main()
{
        char *p;
        printf("%d %d ", sizeof(*p), sizeof(p));

        getchar();
        return 0;
}
```

Output: Compiler dependent. I got output as “1 4”
Explanation:
Output of the above program depends on compiler. sizeof(*p) gives size of character. If characters are stored as 1 byte then sizeof(*p) gives 1.
sizeof(p) gives the size of pointer variable. If pointer variables are stored as 4 bytes then it gives 4.

6. What is the output of this program?

```
#include<stdio.h>
int main()
{
        int x;
        printf("%d",scanf("%d",&x));
        /* Suppose that input value given
                for above scanf is 20 */
        return 1;
```

```
}
```
Output
1
Explanation: scanf returns the no. of inputs it has successfully read.

7. What is the output of this program?
```
#include<stdio.h>
int main()
{
	static int i=5;
	if(--i){
		main();
		printf("%d ",i);
	}
}
```
Output
0 0 0 0
Explanation: Since i is a static variable and is stored in Data Section, all calls to main share same i.

8.What is the output of this program?
```
#include <stdio.h>
int main()
{
	int y = 10;
	if (y++ > 9 && y++ != 10 && y++ > 11)
		printf("%d", y);
	else
		printf("%d", y);
	return 0;
}
```
Output: 13
Explanation : and operator(&) is used so whole expression is evaluated even if the first part is true.

9.What is the output of this program?
```
#include<stdio.h>
int main()
{
	int i = 10;
	i= !i> 14;
	printf("i=%d", i);
	return 0;
}
```
Output: i=0

Explanation : Not oprerator(!) has more precedence than greater than operator(>) so 0>14 is evaluated false.

10.What is the output of this program?

```
#include<stdio.h>
int main()
{
        int a = 3, b = 5, c, d;
        c = a, b;
        d = (a, b);
        printf("c=%d d=%d", c, d);
        return 0;
}
```

Output: c=3 d=5

Explanation : The precedence of '(' is greater as compared to ', ' so firstly a is assigned in c and then b is assigned in d.

11. What is the output of this program?

```
#include<stdio.h>
int main()
{
        char i = 0;
        for (; i++; printf("%d", i))
                ;
        printf("%d", i);
        return 0;
}
```

Output: 1

Explanation : Before entering into the for loop the CHECK CONDITION is "evaluated". Here it is evaluated to 0 (false) and comes out of the loop, and i is incremented (note the semicolon after the for loop).

12.What is the output of this program?

```
#include <stdio.h>
int main()
{
        unsigned char counter = 0;
        for (counter = 0; counter <= 255; counter++) {
                printf("%d ", counter);
        }
        return 0;
}
```

Output: 0 1 2 … infinite times

Explanation: The range of unsigned char is 0 to 255 and when the value of var will cross over 255, value will be 0 and again same process will happen.

13.What is the output of this program?

```
#include<stdio.h>
int fun()
{
static int num = 40;
return num--;
}

int main()
{
for(fun(); fun(); fun())
{
        printf("%d ", fun());
}
getchar();
return 0;
}
```

Output: 38 35 32 29 26 23 20 17 14 11 8 5 2
Explanation: Since num is static in fun(), the old value of num is preserved for subsequent functions calls. Also, since the statement return num– is postfix, it returns the old value of num, and updates the value for next function call.

14.What is the output of this program?

```
#include<stdio.h>
int main()
{
        int i = 20,j;
        i = (printf("Hello"), printf(" All Geeks "));
        printf("%d", i);

        return 0;
}
```

Output: Hello All Geeks 11
Explanation: The printf() function returns the number of characters it has successfully printed. The comma operator evaluates it operands from left to right and returns the value returned by the rightmost expression (See this for more details). First printf(“Hello”) executes and prints “Hello”, the printf(” All Geeks “) executes and prints ” All Geeks “. This printf statement returns 11 which is assigned to i.

15.What is the output of this program?

```
#include <stdio.h>
#define square(x) (x * x)
int main()
{
        int x, y = 1;
        x = square(y + 1);
        printf("%d\n", x);
        return 0;
}
```

Output: 3

Explanation: The macro function square(x)(x*x) calculates the square of the given number.

Initially int y = 1;

In the nextstep x = square(y+1);

x = y+1 * y+1; Here square(x) is replaced x*x.

x = 1+1 * 1+1;

x = 1 + 1 + 1;

x = 3;

16.What is the output of this program?

```
#include <stdio.h>
int main()
{
        int i;
        char input;
        for (i = 1; i<= 5; i++) {
                // The input provided is 'X'
                scanf("%c", &input);
                printf("%c", input);
                ungetc(input, stdin);
        }
        return 0;
}
```

Output: XXXXX

Explanation:The loop will run 5 times. Now, we provide the input as ‘X’. Which is scanned as ‘input’.

printf("%c", input);

The above line prints 'X'

Now the ungetc(input, stdin) statement pushes the character’X’ back into input stream.

For the next run of the loop.

The scanf statement gets the input from “stdin” because of “ungetc” function used previously. Now the printf statement will print ‘X’ Since input= ‘X’ and ungetc(input, stdin) pushes ‘X’ back to the input stream and the same process will take place for the remaining iterations.

17.What is the output of this program?

```
#include <stdio.h>
int main()
{
        int x, a = 0;
        x = sizeof(a++) ?printf("Geeks for Geeks\n") : 0;
        printf("Value of x:%d\n", x);
        printf("Value of a:%d", a);
        return 0;
}
```

Explanation: sizeof is a compile-time operator, so at the time of compilation sizeof and its operand get replaced by the result value. The operand is not evaluated (except when it is a variable length array) at all; only the type of the result matters. In sizeof operator, a++ will not evaluated. So it will remain same i.e. value of a will be 0.
printf returns the number of width. Geeks for Geeks is of 16 width. This will return 16.So x is now 16.

18.What is the output of this program?

```
#include <stdio.h>
int main()
{
        int x;
        x = 5 >8 ?10 : 1 != 2 < 5 ? 20 : 30;
        printf("Value of x:%d", x);
        return 0;
}
```

Output:

Value of x:30
Explanation:
exp1?exp2:exp3
5 >8 ? 10: 1!= 2<5 ? 20:30
Output of exp1 is false, so exp3 (1 != 2 <5 ? 20 : 30) will be evaluated. In exp3, this is also form of ternary operator.
1 != 2< 5 ? 20 : 30 (exp1 ? exp2 : exp3)
Now, exp1 will be evaluated. According to operator precedence, 2<5 will be evaluated first (will give output 1). Now, exp1 is like 1!=1 (condition is false). So, exp3 will be evaluated. Therefore, final output is 30.

19.What is the output of this program?

```
#include <stdio.h>
int main()
{
	unsigned int i = 0x80;
	printf("%d ", i<< 1);
	return 0;
}
```

Output: 256

Explanation :- We know that 0x is hexa-decimal representation of number so 80 converted in decimal is 128 binary(10000000), its left shift 1 so it is (100000000)equal to 256.

20.What is the output of this program?

```
#include <stdio.h>
int main()
{
	typedef struct tag {
		char str[10];
		int a;
	} har;

	har h1, h2 = { "IHelp", 10 };
	h1 = h2;
	h1.str[1] = 'h';
	printf("%s, %d", h1.str, h1.a);
	return 0;
}
```

Output:Ihelp, 10

Explanation: It is possible to copy one structure variable into another like h1 = h2. Hence value of h2. str is assigned to h1.str.

Java OOP

1. What is the output of this program?

```
class Writer
{
	public static void write()
	{
		System.out.println("Writing...");
	}
}
class Author extends Writer
{
	public static void write()
```

```
	{
		System.out.println("Writing book");
	}
}

public class Programmer extends Author
{
	public static void write()
	{
		System.out.println("Writing code");
	}

	public static void main(String[] args)
	{
		Author a = new Programmer();
		a.write();
	}
}
```

Output: Writing book
Explanation: Since static methods can't be overridden, it doesn't matter which class object is created. As a is a Author referenced type, so always Author class method is called. If we remove write() method from Author class then Writer class method is called, as Author class extends Writer class.

2.What is the output of this program?

```
class Derived
{
	public void getDetails()
	{
		System.out.printf("Derived class ");
	}
}

public class Test extends Derived
{
	public void getDetails()
	{
		System.out.printf("Test class ");
		super.getDetails();
	}
	public static void main(String[] args)
	{
		Derived obj = new Test();
		obj.getDetails();
```

```
        }
}
```
Output: Test class Derived class
Explanation: super keyword is used to invoke the overridden method from a child class explicitly.

3. What is the output of this program?
```
class Alpha
{
        static String s = " ";
        protected Alpha()
        {
                s += "alpha ";
        }
}
class SubAlpha extends Alpha
{
        private SubAlpha()
        {
                s += "sub ";
        }
}

public class SubSubAlpha extends Alpha
{
        private SubSubAlpha()
        {
                s += "subsub ";
        }
        public static void main(String[] args)
        {
                new SubSubAlpha();
                System.out.println(s);
        }
}
```

Output:

alpha subsub
Explanation: SubSubAlpha extends Alpha! Since the code doesn’t attempt to make a SubAlpha, the private constructor in SubAlpha is okay.