1. A function prototype is used for
   a) Declaring the function logic
   b) Calling the function from the main body
   c) Telling the compiler, the kind of arguments used in the function
   d) Telling the user for proper use of syntax while calling the function

Solution: (c) A function prototype tells the compiler what kind of arguments a function is looking to receive and what kind of return value a function is going to give back. This approach helps the compiler ensure that calls to a function are made correctly and that no erroneous type conversions are taking place.

2. What is the output of the following C program?

```
#include <stdio.h>
void foo(), f();
int main()
{
  f();
return 0;
}
void foo()
{
   printf("2 ");
}
void f()
{
   printf("1 ");
   foo();
}
```

a) Compiler error as foo() is not declared in main
b) 1 2
c) 2 1
d) Compile time error due to declaration of functions inside main

Solution: (b) The function f() is called from the main. After printing ‘1’, foo() will be called and ‘2’ will be printed. This is an example of nested function.

3. How many times ‘Hi’ will be printed in the program given below

```
#include<stdio.h>
int i;
int fun();
```

```
int main()
{
   while(i)
   {
      fun();
      main();
   }
   printf("Hello\n");
   return 0;
}
int fun()
{
   printf("Hi");
}
```

a) Only once
b) Zero times
c) Infinite times
d) Compilation error

Solution: (b) The default value of i is ‘0’. Thus, the while loop will never be executed and the control will not come into the function. Thus, ‘Hi’ will never be printed.

4. What is the output of the C code given below

```
#include <stdio.h>
float func(float age[]);

int main()
{
   float result, age[] = { 23.4, 55, 22.6, 3, 40.5, 18 };
   result = func(age);
   printf("%0.2f", result);
   return 0;
}

float func(float age[])
{
   int i;
   float result, sum = 0.0;
   for (i = 0; i < 6; ++i) {
```

```
        sum += age[i];
    }
    result = (sum / 6);
    return result;
}
```

a) 27.08
b) 27.083334
c) Compiler error as result is declared twice
d) Error: Invalid prototype declaration

Solution: (a) The program finds the average of the array elements. As the variables declared in the function are local, they can be used in the main as well. 0.2f will print two decimal points. Thus the output is 27.08.

5. Which statement is correct about Passing by value parameters?

a) It cannot change the actual parameter value
b) It can change the actual parameter value
c) Parameters is always in read-write mode
d) None of them

Solution: (a) It cannot change the actual parameter value

6. What will be the output?

```
int main()
{
    {
        int a = 70;
     }
     {
        printf("%d", a);
     }
  return 0;
}
```

a) 70
b) Garbage value
c) Compilation error
d) None

Solution: (c) Compilation error.

A Block is a set of statements enclosed within left and right braces ({and} respectively). A variable declared in a block is accessible in the block and all inner blocks of that block, but not accessible outside the block. Thus the program produces compiler error.

7. How many times Hello world will be printed?

```
#include<stdio.h>
int main()
{
   printf("Hello world\n");
   main();
   return 0;
}
```

a) Infinite times
b) 32767
c) 65535
d) Till stack overflows

Solution: (d) Till stack overflows

A call stack or function stack is used for several related purposes, but the main reason for having one is to keep track of the point to which each active subroutine should return control when it finishes executing.
A stack overflow occurs when too much memory is used on the call stack. Here function main() is called repeatedly and its return address is stored in the stack. After stack memory is full. It shows a stack overflow error.

8. What will be the output?

```
#include<stdio.h>
void func(int n, int sum)
{
int k = 0, j = 0;
if (n == 0) return;
        k = n % 10;
j = n / 10;
sum = sum + k;
func (j, sum);
printf ("%d, ", k);
}

int main ()
{
int a = 2048, sum = 0;
func (a, sum);
printf ("%d ", sum);
return 0;
}
```

a) 8 ,4, 0, 2, 14
b) 8, 4, 0, 2, 0
c) 2, 0, 4, 8. 14
d) 2, 0, 4, 8, 0

Solution: (d) 2, 0, 4, 8, 0
sum has no use in func(), it is there just to confuse. Function func() just prints all digits of a number. In main, there is one more printf statement after func(), so one more 0 is printed after all digits of n.

9. Consider the function

```
find(int x, int y)
{
        return((x<y) ? 0 : (x-y));
}
```

Let a and b be two non-negative integers. The call find(a, find(a, b)) can be used to find the

a) Maximum of a, b
b) Positive difference between a and b

c) Sum of a and b
d) Minimum of a and b

Solution: (d) Minimum of a and b

The function returns

0 if x<y

x-y if x>y

Assume in first case a>b so find(a,b) will return (a-b). So find(a,find(a,b)) will be find(a,a-b) which will return (a-(a-b))= b which is the minimum number.

Now if a<b then find(a,b)=0 so find(a,find(a,b))=find(a,0) which will return (a-0)=a which is the minimum number. So, the code actually returns the minimum of two numbers.

10. Consider the function

```
int fun(x: integer)
{
if x > 100 then fun = x – 10;
else
fun = fun(fun(x + 11));
}
```

For the input x = 95, the function will return

a) 89
b) 90
c) 91
d) 92

Solution: (c)

f(95)–>f(f(106)) = f(96)–>f(f(107)) = f(97)–>f(f(108)) = f(98)–>f(f(109)) = f(99)–>f(f(110)) = f(100)–>f(f(111))=f(101)=91

So the correct option is (c)