Week 8: Assignment 8 Solution

1. What is the default return type if it is not specified in the function definition?
   a) void
   b) int
   c) double
   d) float

Solution: (b) Integer is the default data type if not specified in the function.

2. What will be the output?

```
#include<stdio.h>
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.

3. How many times will ‘Hello world’ 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

Week 8: Assignment 8 Solution

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 the stack memory is full. It shows a stack overflow error.

4. Which of the following statements is correct?
   a) A function with same name cannot have different signatures
   b) A function with same name cannot have different return types
   c) A function with same name cannot have different number of parameters
   d) All of the mentioned

Solution: (d) All of the statements mentioned are correct in C.

5. What will the function return?

```
int func(int x, int y)
{
  if (y==0) return 0;
  if (y ==1) return x;
  return x+func(x, y-1);
}
```

   a) x * y where x and y are integers
   b) x * y where x and y are non-negative integers
   c) x + y where x and y are integers
   d) x + y where x and y are non-negative integers

Solution: (b) x * y where x and y are non-negative integers

6. 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);
}
```

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.

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

```
#include <stdio.h>
int fun(int n)
{
   int i, j, sum = 0;
   for(i = 1;i<=n;i++)
      for(j=i;j<=i;j++)
         sum=sum+j;
   return(sum);
}
int main()
{
   printf("%d", fun(5));
   return 0;
}
```

a) 25
b) 5
c) 15
d) 10

Solution: (c) The program finds the sum of the integer numbers till 5. Thus the output is 15.

8. What will be the output?

```
#include <stdio.h>
void swap(int a, int b)
{
   int temp;
   temp = a;
   a = b;
   b = temp;
}

int main()
{
   int num1 = 10, num2 = 20;
   printf("Before swapping num1 = %d num2 = %d\n", num1, num2);
   swap(num1, num2);
   printf("After swapping num1 = %d num2 = %d \n", num1, num2);
   return 0;
}
```

a) Before swapping num1 = 10 num2 = 20
   After swapping num1 = 10 num2 = 20

b) Before swapping num1 = 10 num2 = 20
   After swapping num1 = 20 num2 = 10

c) Before swapping num1 = 10 num2 = 20
   After swapping num1 = 20 num2 = 20

d) Before swapping num1 = 10 num2 = 20
   After swapping num1 = 10 num2 = 10

Solution: (a) Before swapping num1 = 10 num2 = 20
After swapping num1 = 10 num2 = 20

In function swap, receive the parameters through variables a and b respectively. Copied the value of variable a to the variable temp. Copied the value of variable b to the variable a and copy the value of variable temp to the variable b. This will do the swapping ONLY in the swap() function, but it will NOT change the value of variables in the main() function.

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 the 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.

Week 8: Assignment 8 Solution

10.

```
int fibonacci (int n)
{
switch (n)
{
default:
return (fibonacci(n - 1) + fibonacci(n - 2));
case 1:
case 2:
}
return 1;
}
```

The function above has a flaw that may result in a serious error during some invocations.
Which one of the following describes the deficiency illustrated above?

(a) For some values of n, the environment will almost certainly exhaust its stack space before the calculation completes.
(b) An error in the algorithm causes unbounded recursion for all values of n.
(c) A break statement should be inserted after each case. Fall-through is not desirable here.
(d) The fibonacci() function includes calls to itself. This is not directly supported by Standard C due to its unreliability.

Solution (a) For some values of n, the environment will almost certainly exhaust its stack space before the calculation completes.

The function calls itself. So it's an example of recursion. As there is no proper terminating condition for the function, for a large value of n it will exhaust the storage space (stack) which will stop the execution without final result.