BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI (RAJ.)

Second Semester 2019-20 CS F111 Computer Programming

LABORATORY SESSION #8

(*Flow Control-II, Functions*)

1. Write a menu-driven program which allows the user to repeatedly enter a choice and a number.
   a. If the choice is **b**, the number entered is binary (unsigned). Write a function to convert it to decimal and return the corresponding value to main().
   b. If the choice entered is **d**, the number entered is decimal. Write a function to convert it to binary and return the corresponding value to main().
   c. If the choice entered is **x**, the program terminates.
   d. If any other choice is entered, ask the user to enter a valid choice.
   e. For both the functions above, you need to think of the appropriate return type and the number/type of arguments to be passed.

2. An Armstrong number is an n-digit number such that the sum of its digits raised to the power n is the number itself. For example, the number 153 has three digits, and $1^3+5^3+3^3 = 153$. Therefore 153 is an Armstrong number.
   Write a C program which accepts two numbers and finds all Armstrong numbers in that range. You should create a function IsArmstrong() for checking whether a number is an Armstrong number or not, and call it repeatedly from within main().

3. Read the following C program and understand it. What is the purpose of `my_fun()` in this program? Is the function `my_fun()` able to achieve its intended functionality. Write down your observation in your note book with proper justification.

```
#include <stdio.h>
/* function declaration */
void my_fun(int x, int y) {
   int z;
   z = x;   x = y;      y = z;
   return;
}
int main () {
int a = 100;   int b = 200;
 printf("Value of ‘a’ before my_fun()
    execute : %d\n", a );
 printf("Value of ‘b’ before my_fun()
    execute : %d\n", b );
my_fun(a, b);
 printf("Value of ‘a’ before my_fun()
    execute : %d\n", a );
 printf("Value of ‘b’ before my_fun()
    execute : %d\n", b );
return 0;       }
```

4. You are now familiar with the hailstone sequence for generation, of which you had written a C program earlier (refer to the 6th question of Lab Sheet #7). For today's lab, instead of generating and printing the hailstone sequence inside the `main()`, write a function whose prototype is given below, and call the function from within main():

```
int printHailstones(int val); /* prints all hailstones starting from val,
                              and returns the number of terms found in the
                              sequence */
```

(Hint: All you need to do is to simply copy and paste the existing code into the new function, count the terms and return it, and include the function call inside `main()` */

5. What would be the output of each of the following pieces of code/functions? First attempt answering the question yourself, and then try and check your answer by compiling the programs.

a.
```
#include<stdio.h>
void func(int a,int b);
int main(void)
{     int x;
      x=func(2,3);
      return 0;
}
void func(int a,int b)
{     int s;
      s=a+b;
      return;
}
```

b.
```
#include<stdio.h>
int diff(int x,int y)
{  return x-y; }
int sum(int x,int y)
{  return x+y; }
int main(void)
{
  int a=20, b=5, c=2, d=6;
  printf("%d\t", a + diff(d,c));
  printf("%d\n", diff(a,sum(diff(b,c),d)));
  return 0;
}
```

c.
```
#include<stdio.h>
void func(void);
int main(void)
{     int i=5;
      for(i=i+1; i<8; i++)
            func();
}
void func(void)
{     int j;
      for(j=1; j<3; j++)
            printf("%d\t",++j);
}
```