Birla Institute of Technology & Science, Pilani
Computer Programming (CS F111)
Second Semester 2014-2015
Lab Sheet on Functions

Objectives

1. Modular Programming: Functions in C
2. Passing Arguments to Functions
3. Passing Arrays to Functions
4. Exercises

Modular Programming (Top down Approach):

The concept of modular programming is to divide the given big problem in small sub problems, solve these smaller sub problems, and combine the solutions of these smaller sub problems to get the solution of big problem. The C language support modular programming through ***functions***.

A ***function*** is a named, independent section of C code that performs a specific task and optionally returns a value to the calling program. The small sub-problems can be implemented through functions in C and the solution to these sub-problems are the return values of these functions which are returned back to the calling function (usually the main() function in C).

The general syntax of defining a function is:

```
Return_Type  Name_of_Function (List of parameters)
{
  // Function Body
}
```

Type the following program and observe how it is working:

```
void message(); //Function Declaration
#include<stdio.h>
int main()
{
  printf("I am in function main()\n");
  message(); /* Function message() is called in function main() */
  printf("I return back from the function message()\n");
  return 0;
}
void message() /* Function to print a message. It does not return
                  anything and does not take any parameters as input*/
{
      printf("I am in function message\n");
       return;
}
```

Explanation: *All C program execution starts from function main(), After printf statement, there is one function call statement for the function named message(). So the program control will transfer to function message() and it will execute the statements of this function. From the return statement of*

function message(), the control will return back to the statement which is immediately written after the function call in main(). Then it will execute the rest of the statements in main().Since the function is not returning any value, its return type is void.

Example 1

Consider the following problem to find the sum of sin(x) series i.e**. $x - x^3/3! + x^5/5! -$**… Here the problem is divided into two sub problems: one is to find the factorial and another is to calculate the sum of sin(x) series using factorial. Finally in function main the calculated sum is displayed.

Type the following code. Observe and understand the output.

```
long int fact( int n)/* Function Definition */
{
 int i;
 long int f=1;
 for( i=1; i<=n; i++ ) f *= i;
 return f;
}

float sinsum(float x, int n) /* Function Definition */
{
   int i,sign=1;
   float sum=0.0, element;
   for( i=1; i<=n; i++ ) {
      element = sign * pow(x,2*i-1)/fact(2*i-1); /* Function Call */
      sign *= -1;
      sum += element;
   }
   return sum;
}

/* sinsum is taking two arguments x and n, where x is of type float
and n is of type integer */

#include<stdio.h>
#include<math.h>

long int fact(int n); /* Function Declaration */
float sinsum(float x, int n); /*Function Declaration */

int main()
{
  int n;
  float x, sin_sum;
```

```
  printf("Enter value of x and n");
  scanf("%f %d",&x,&n);
  sin_sum = sinsum(x,n); /* Function Call */
  printf("Sum of sin series = %f",sin_sum);
return 0;
}
```

Explanation: *The function fact() takes one input parameter and returns the value. Similarly, sinsum() function takes two input parameters and returns the resultant sum of series. The data type of input arguments in function definition should match as in call to function. Also the data type of the variable which is returned back to the calling function should match return type of the function and the variable in the calling function which holds it. For example, variable sin_sum is declared as float in main which is same as variable sum in sinsum() function. Here the input parameters are* ***passed by value****.*

Exercise: Modify the above program to print the sum of following series:

a) $x + x^2/3! - x^3/3! + \ldots$

b) cosine series

Passing one dimensional array to a function:

Type the following code. Observe and understand the output.

```
void readArray(int a[],int size); // Observe how array a[] is passed as input parameter
void printArray(int a[],int size);
int arraySum(int a[], int size);
int main()
{
      int a[20],sum,n;
      printf("Enter number of elements in array a:\n");
      scanf("%d",&n);
      readArray(a,n);
      printArray(a,n);
      sum = arraySum(a,n);
      printf("sum = %d\n",sum);
}
void readArray(int a[],int size){
      int i; //local variable: local to function readArray()
      for(i=0;i<size;i++)
          scanf("%d",&a[i]);
}
void printArray(int a[],int size){
      int i;
      for(i=0; i<size; i++)
          printf("%d",a[i]);
}
int arraySum(int a[], int size){
      int i,sum; // i and sum are local to function arraySum()
```

```
      for(i=0; i<size; i++)
            sum = sum + a[i];
      return sum;
}
```

Explanation: *Size of array should be passed along with array name to a function. Array name contains the base address (or address of first element) of the array. So with the help of array name and array index corresponding array element can be accessed.*

Exercises

1. Write a C function named to calculate the roots of a quadratic equation. Call the function in main() to test it.

2. Write a C function that takes an integer number as a parameter and returns the reverse of it. Call this function in main().
3. Write a program that reads integers in an integer array. Write functions to accomplish the following tasks on the array elements:
   (i) Sort the array using bubble sort algorithm.
   (ii) Find requested integer using binary search algorithm.
   Call both the functions in main().

4. Write a program that reads and maintains the price list of books in one dimensional array of type float in increasing order of price. The program should do following operations on the price list in separate functions (write separate functions for these tasks and call them in main()):
   (i) Insert a new book's price into the list without changing the order. [Hint: Require shifting of elements right after insertion.]
   (ii) Find and delete the requested price entry from the list. [Hint: Require shifting of elements left after finding the element to be deleted.]
   (iii) Count number of books whose price is less than Rs.150.
   (iv) Count number of books whose price is more than Rs. 1550.

**********Enjoy Coding **********