Programming and Data Structures Laboratory | 2021-22 Autumn semester, Section 20
Assignment 5 | January 18, 2022

Submission instructions

* Submit one .c file for each part of the assignment. Each of your .c files should be named as:

<your roll number>_A<assignment number>_<part number>.c

For instance, if your roll number is 21CS10023, and if you are presently doing Assignment 5 which has 3 parts, then you should submit 4 separate .c files named as:
21CS10023_A5_1.c 21CS10023_A5_2.c 21CS10023_A5_3.c

***** Submissions must be through the course Moodle, before the end of Lab session (11:55 AM). Late submissions will be penalized / not accepted.
***** Try to think how you can solve a problem efficiently. A program that solves the problem efficiently will get more credit.

---

1. [15 Marks] Write a c program where two sorted arrays A, B are taken as input from the user in the main function. Write a function 'merge_array' that takes in two sorted arrays and merges those two arrays to get a merged sorted array. From the main function, invoke the 'merge_array' function, pass both the arrays to it. Print that merged sorted array from the main function. Next take another sorted array C as input in main. Again invoke the 'merge_array' function which takes the previously merged (sorted) array and array C as input and merges them to get a new sorted array. Print the finally sorted array from the main function. You can take the max size of each of the user supplied arrays to be 10 and assume the user supplies the number of elements of each array before supplying the array elements.
Example output:

```
Enter the number of elements in the first array (A): 3
Enter the elements of the first array (A): 4 7 12
Enter the number of elements in the second array (B): 4
Enter the elements of the second array (B): 5 6 7 10
Merged array: 4  5  6  7  7  10  12
Enter the number of elements in the third array (C): 4
Enter the elements of the third array (C): 2 3 7 8
Merged array: 2  3  4  5  6  7  7  7  8  10  12
```

2. [15 Marks] Write two functions "int reverse(int N)" and "int reverse_recursive(int N)" both of which takes a decimal integer N as argument and returns the integer which is formed by reversing the order of digits of N. In "int reverse(int N)", do not use recursion to get the reversed number while in "int reverse_recursive(int N)" recursively call the function to get the reversed number. If required, you can use a global variable [that can be used throughout the program]. Write a main() function to demonstrate the use of both the functions.
Example output:

```
Please provide the positive integer to be reversed: 1234
The reverse of 1234 using "reverse_recursive" is: 4321
The reverse of 1234 using "reverse" is: 4321

Please provide the positive integer to be reversed: 7890
The reverse of 7890 using "reverse_recursive" is: 987
The reverse of 7890 using "reverse" is: 987
```

3. [20 Marks] For any real number x, the series expansion of sin(x) is as follows:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Write a program that takes an user given real number x [where x (in degrees) is the value of the angle] and computes the value of sin(x) up to a precision of 0.0001, that is the actual value of sin(x) and the value of sin(x) computed by the above formula, should not differ by a factor of more than 0.0001. You should write a function "sine_series" which takes the angle value supplied by the user. The return type and the argument types have to be properly set. The function should compute the value of the sin of the angle using the formula as said above and return the computed value. Call the function from main and print the value of sin of x from main. You can use math.h library functions to get the actual value of sin(x) which you will need to compare to the computed value of sin(x) upto a precision of 0.0001.

Example output:

```
The value of x (in degrees): 45
The value of sin45.000000 is 0.707215

The value of x (in degrees): 30
The value of sin30.000000 is 0.500061
```