Section 14

PDS Lab **Lab - 6** **28.12.2022**

Instructions:

Give sufficient comment against each statement in your program.
You should save each program with the file name as specified against each problem.
There is a partial credit even if your program does not run successfully for all the test cases as mentioned.
You should upload all the programs (with *.zip*) before leaving the lab session. No Moddle submission will attract zero credit in the evaluation.

1. Write a C program that in the main function would prompt the user to enter the number of the terms (say k) of the following series he/she wants to be displayed. Write a function named **print_term** to compute the nth term of the series, using recursion. Based on the user's input, call the function **print_term** in the main function in a loop with appropriate argument to print the first k terms of the following series:

$$t(n) = 3 * t(n-1) * t(n-2) - 2 * t(n-2) * t(n-3) + 1,$$
$$Given, t(0) = 0, t(1) = 1, t(2) = 2.$$

Test Case:

#	INPUT	OUTPUT
1	k=3	0,1,2
2	k=-1	Invalid input
3	k=4	0,1,2,7
4	k=6	0,1,2,7,39,792

[Time: 25 Minutes] [20]

2. Two numbers are said to be co-prime, if the greatest common divisor of the numbers is one. For examples, 13 and 14 are co-prime but 14 and 21 are not. Write a C function ***void CoPrime (int a, int b)*** to test whether the pair of numbers *a* and *b* are co-prime. In the main program, read, say *n* numbers and use this function to test how many pair of them are co-prime.

[Hint: You should define ***int gcd (int m, int n)*** to find the greatest common divisor of two numbers and ***void pair(int arrayName)*** to output all the pairs from a given set of numbers. Both the function should be defined **recursively.**]

Test Case:

#	INPUT	OUTPUT
1	[11 15 20 21 25]	Co-prime pairs: (11 15) (11 20) (11 21) (11 25) (20 21) (21 25)
2	[11.5 15 20.8 21 25]	Invalid entries: All should be integer numbers
3	[11 -15 20 -21 25]	Invalid entries: All should be positive numbers
4	[18 35 31 25 7]	Co-prime pairs: (18,35) (18,31) (18,25) (18,7) (35,31) (31,25) (31,7) (25,7)

[Time: 25 Minutes] [20]

3. Write a C function ***int* Merge(size1, size2, *ptr1, *ptr2)*** to merge two sorted arrays and return the resulting sorted array. Here, ***size1*** and ***size2*** represent the number of elements of the two arrays and ****ptr1, *ptr2*** points to two given sorted arrays.

- In the main program:
  - Read the two sorted arrays with their sizes from the key
  - Merge them into one sorted array.
  - Print the input sorted arrays
  - Print the Resulting sorted array after merging in the main function.
- Example:
  - Two Input arrays
    - A = (5, 11, 17, 33, 67, 98)

      B = (2, 20, 21, 45,76, 100)
- Output Array

  C = (2, 5, 11, 17, 20, 21, 33, 45, 67, 76, 98, 100)

Test Case:

#	INPUT	OUTPUT
1	Enter size: 3 Enter values: 1 2 3 Enter size: 4 Enter Values: 2 4 5 6	A = 1 2 3 B = 2 4 5 6 C = 1 2 2 3 4 5 6
2	Enter size: 2 Enter values: 5 10 Enter size: 2 Enter Values: 1 2	A = 5 10 B = 1 2 C = 1 2 5 10
3	Enter size: 3 Enter values: 1 2 3 Enter size: 3 Enter Values: 1 2 3	A = 1 2 3 B = 1 2 3 C = 1 1 2 2 3 3
4	Enter size: 5 Enter values: -5 -3 0 1 2 Enter size: 1 Enter Values: 3	A = -5 -3 0 1 2 B = 3 C = -5 -3 0 1 2 3

[Time: 20 Minutes] [3+8+4=15]

4. A set can be represented by a dynamic array of elements, where no repetition is permitted.

Write C functions to perform the following operations on sets of integer valued elements.

- ***int* BuildSet(int n)***:
  - Read n numbers of elements and store them in a set.
- ***int SearchSet(int* A, int x):***
  - Search the set A to find if an element x is in it.
- Given two sets, compute the following.
  - ***int* Union (int *A, int *B):*** To return the union of two sets A and B.
  - ***int* Intersection (int *A, int *B):*** To return the intersection of two sets A and B.
  - ***int* Difference (int *A, int *B):*** To return the difference of two sets A and B.

The main program should call these functions.

All results should be stored in their resultant sets and then display the results.

[**Hint:** Define a function ***void Print(int *A)*** to print a set A.

- In the main method:
  - Call BuildSet() for set A
  - Call BuildSet() for set B
  - Show a menu to user:
    - Press '1' to perform search in Set A
    - Press '2' to perform search in Set B
    - Press '3' to perform Union of Set A and Set B
    - Press '4' to perform Intersection of Set A and Set B
    - Press '5' to perform Difference of Set A and Set B
    - Press any other key to quit
  - The menu should run in an infinite loop

Test Case:

#	INPUT	OUTPUT
1	A = 1 2 3 4 5, B = 2 3 4 5 1 5 2 6	5 is present in set A 6 is NOT present in set B Union: 1 2 3 4 5

#	INPUT	OUTPUT
	3 4 5 0	Intersection: 2 3 4 5 Difference: 1
2	A = 1 2 3, B = 4 5 6 1 5 2 6 3 4 5 0	 5 is NOT present in set A 6 is present in set B Union: 1 2 3 4 5 6 Intersection: Null Difference: 1 2 3

[Time: 90 Minutes] [5+5+10+10+10+5 = 45]

---*---