Section 14

PDS Lab **Lab - 5** **07.12.2022**

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.
No Moddle submission will attract zero credit in the evaluation.
Name the files as {ROLL}_A{#}_Q{#}.c, without the { and }. For ex: 19CS91R05_A2_Q1.c
Consult your TA for any confusion. Penalty if the file names do not stick to this convention.

1. Find the sum of the first *n* terms of the following series.

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Create a *int* ***power (int x, int n)*** function to compute x^n and i*nt* ***fact (n)*** function to compute the factorial *n!* for a given number n for this problem. Values of *x* and *n* are inputs given by user. Do not use the factorial or power functions available in `math.h`. Assume: *x* is an integer value and n≤10.

#	INPUT	OUTPUT
1	10 1	1.0
2	0 10	1.0
3	1 10	2.7183
4	2 5	7.0

[Time: 25 Minutes] $[3 + 4 + (2 \times 4)=15]$

2. Two numbers are said to be co-prime, if the greatest common divisor (GCD) of the numbers is one.
For example,

- 13 and 14 are co-prime
- 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 five numbers and store in an array of integers.
- Use CoPrime() function to test how many pair of them are co-prime.

[Hint: you should also define ***int gcd(int a, int b)*** to find the greatest common divisor of two numbers and ***void pair(int a[])*** to find all the pairs from a given set of numbers stored in the array a]

#	INPUT	OUTPUT
1	13 14 15 16 17	13 and 14 are Co-Prime 13 and 15 are Co-Prime 13 and 16 are Co-Prime 13 and 17 are Co-Prime 14 and 15 are Co-Prime 14 and 17 are Co-Prime 15 and 16 are Co-Prime 15 and 17 are Co-Prime 16 and 17 are Co-Prime
2	2 3 4 5 6	2 and 3 are Co-Prime 2 and 5 are Co-Prime 3 and 4 are Co-Prime 3 and 5 are Co-Prime 4 and 5 are Co-Prime 5 and 6 are Co-Prime
3	2 4 6 8 10	No Co-Prime found
4	5 5 5 5 5	No Co-Prime found

[Time: 35 Minutes] [5+5+5+(2.5× 4) = 25]

3. Write a C program to make a ***Guessing game*.**

The objective of the game is to guess a random number between 1-100 in 5 or less tries.

- Your game should initialize a random number, say *r*.
  [Hint:
  - use `rand()%100+1` to get a random number *r* between 1 and 100, also use
  - use `srand(time(0));` (at the start of the main method)
  - use `#include <stdlib.h>`
  - and use `#include <time.h>`

  ]
- Your game will ask the user to guess a number between 1 – 100, say *g*.
  - If the answer is correct:
    - Display "Winner" and terminate the program.
  - Else
    - If the user entered number, *g*, is greater than *r*,
      Display "Your guess is too big!"
    - Else, Display "Your guess is too small!"
- Repeat the above process (maximum 5 times).
  - If the user fails to guess the correct number in their 5th attempt.
    Display: "Loser: The number was: %d",r"

[Assume the user always enter a valid number between 1-100]

#	INPUT	OUTPUT
1	50 25 12 6 2	Your guess is too big. Your guess is too big. Your guess is too big. Your guess is too big. Your guess is too big. Loser: The Number was 1
2	50 75 87 95 98	Your guess is too small. Your guess is too small. Your guess is too small. Your guess is too small. Your guess is too small. Loser: The Number was 100
3	50 85 65 70	Your guess is too small. Your guess is too big. Your guess is too small. Winner.
4	50	Winner.

[Time: 40 Minutes] [5+5+5+5+5 = 25]

4. Write a program to do the following:
   - Consider an integer array a = {a_1, a_2, ... , a_n}.
   - Suppose the elements are in the range [r_1, r_2]
   - We wish to see the distribution of array elements in some ranges.
   - To do this, we will create "bins" that
     - Divides the entire range of values into a series of
       - Consecutive, Equal, Non-overlapping intervals and
     - then count how many values fall into each interval.
   - The number of elements of each bin may differ from each other.

   For example, consider an array

   - a[] = {10, 1, 14, 5, 22, 51, 46, 37, 9, 27, 55, 49, 72, 24, 47, 4, 67, 30, 40, 15}.
   - Number of bins = 4
   - Since the elements have range from 1 to 72, i.e. [1,72]
     the bins/intervals will be the following:
     - 1-18,
     - 19-36,
     - 37-54,
     - 55-72.
   - The elements of each bin will be:
     - bin1-> 10, 4, 14, 5, 9, 1, 15
     - bin2-> 22, 27, 24, 30
     - bin3-> 51, 46, 37, 49, 47, 40
     - bin4-> 55, 72, 67
   - Your task is to take the array elements and the number of bins from the user. Then
     - create bins and
     - put the array elements in appropriate beans,
     - Also, output the bin contents as well as number of elements in each bin.

   Write suitable functions. Comment your code appropriately.

#	INPUT	OUTPUT
1	N = 10 A[N] = 12, 29, 20, 3, 9, 11, 26, 17, 4, 19 Bins = 3	bin1-> 3, 9, 11, 4 Elems = 4 bin2-> 12, 20, 17, 19 Elems = 4 bin3-> 29, 26 Elems = 2
2	N = 10 A[N] = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Bins = 3	bin1-> 1, 2, 3 Elems = 3 bin2-> 4, 5, 6 Elems = 3 bin3-> 7, 8, 9, 10 Elems = 4
3	N = 10 A[N] = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Bins = 1	bin1-> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Elems = 10
4	N = 10 A[N] = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 Bins = 10	{10 bins with one element each}

[Time: 50 Minutes] [10+15+10=35]