Section 6 Autumn-2024

PDS Lab **Lab-7** **01.10.2024**

Instructions:

- This lab is based on the topics: Functions.
- You should save each program with the file name as specified against each problem as <Lab#>-<Assignment#>-<Roll#>.c. For example, **06-01-24NA10006.c** to save Program to 1st assignment in Lab 6 with Roll Number 24NA10006
- You should upload each program to the Moodle system. Also, copy + paste your programs to the text window on the test page.
- A few test cases against each problem are given for your reference, including but not limited to.
- There are three problems and the maximum time allowed is 120 minutes.
- **Do not use pointers and recusrion in this lab.**

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

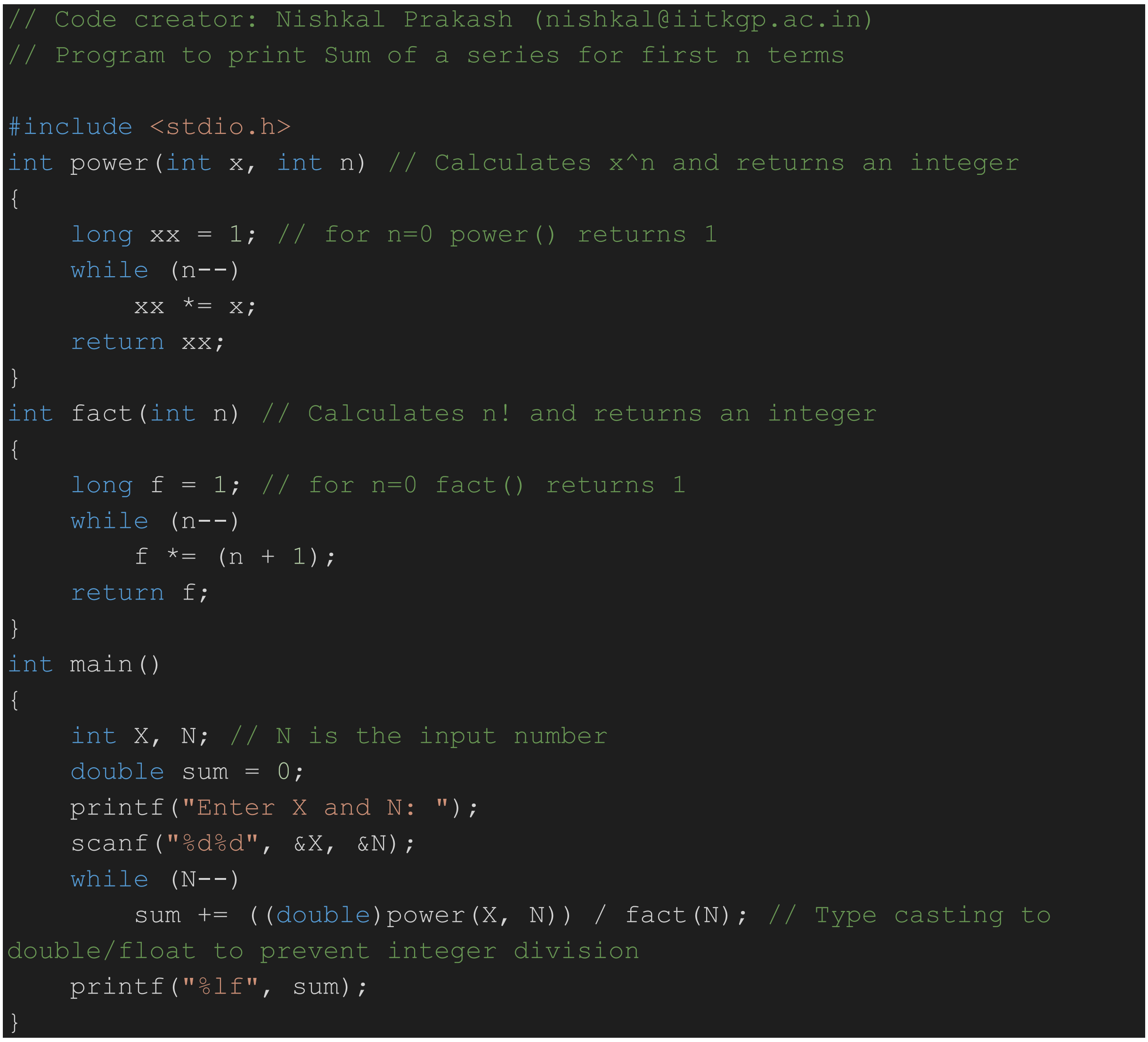
$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

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.

Test cases:

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

$[6 + 6 + (3 \times 4) + 6 = 30]$



```c
// Code creator: Nishkal Prakash (nishkal@iitkgp.ac.in)
// Program to print Sum of a series for first n terms

#include <stdio.h>
int power(int x, int n) // Calculates x^n and returns an integer
{
    long xx = 1; // for n=0 power() returns 1
    while (n--)
        xx *= x;
    return xx;
}
int fact(int n) // Calculates n! and returns an integer
{
    long f = 1; // for n=0 fact() returns 1
    while (n--)
        f *= (n + 1);
    return f;
}
int main()
{
    int X, N; // N is the input number
    double sum = 0;
    printf("Enter X and N: ");
    scanf("%d%d", &X, &N);
    while (N--)
        sum += ((double)power(X, N)) / fact(N); // Type casting to
double/float to prevent integer division
    printf("%lf", sum);
}
```

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,

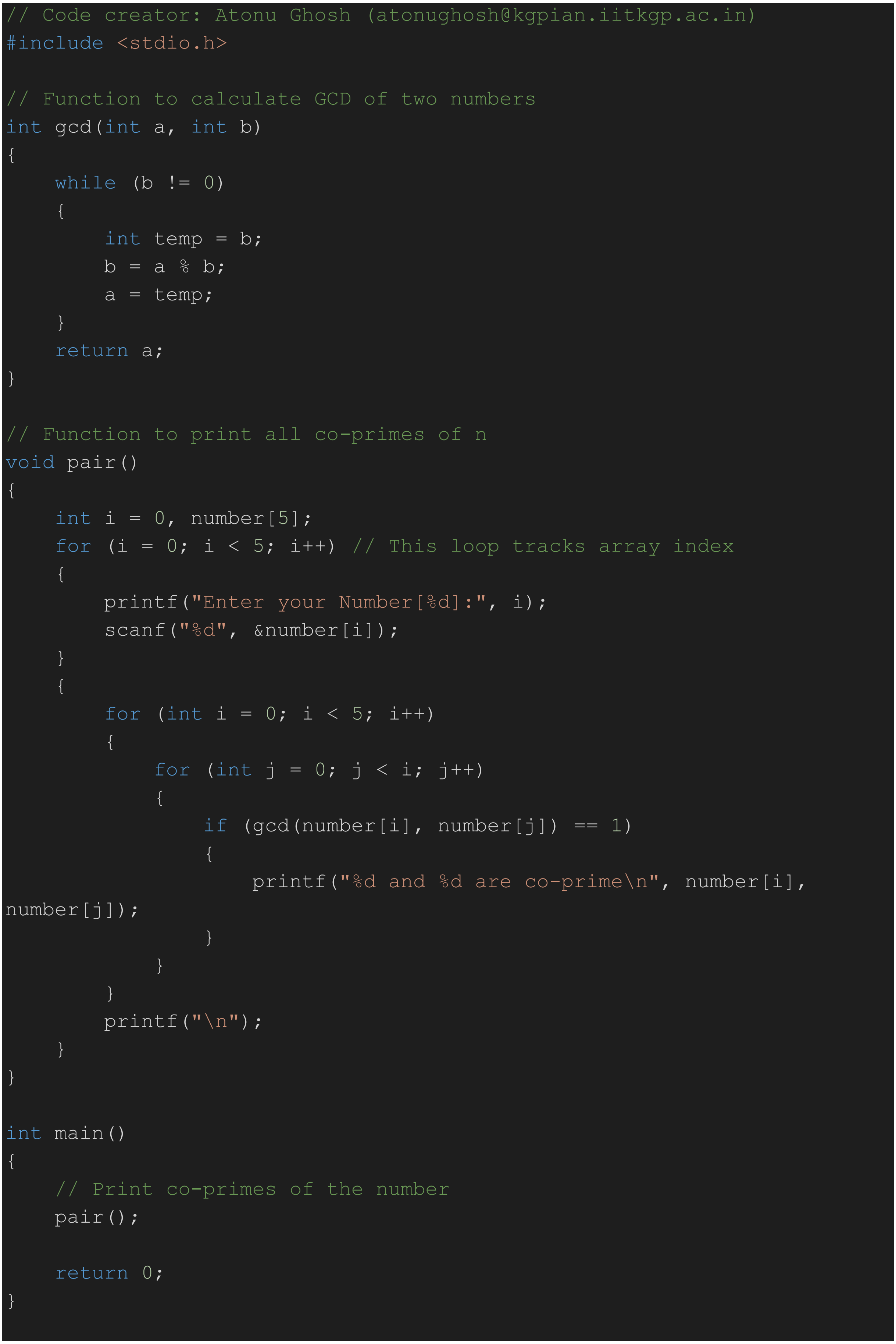
- Read five numbers and store them in an array of integers.
- Use the CoPrime() function to test how many pairs of them are co-prime.

[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]

Test cases:

#	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

$[5+5+5+(2.5\times 4) + 5 = 30]$



```
// Code creator: Atonu Ghosh (atonughosh@kgpian.iitkgp.ac.in)
#include <stdio.h>

// Function to calculate GCD of two numbers
int gcd(int a, int b)
{
    while (b != 0)
    {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// Function to print all co-primes of n
void pair()
{
    int i = 0, number[5];
    for (i = 0; i < 5; i++) // This loop tracks array index
    {
        printf("Enter your Number[%d]:", i);
        scanf("%d", &number[i]);
    }
    {
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < i; j++)
            {
                if (gcd(number[i], number[j]) == 1)
                {
                    printf("%d and %d are co-prime\n", number[i],
number[j]);
                }
            }
        }
        printf("\n");
    }
}

int main()
{
    // Print co-primes of the number
    pair();

    return 0;
}
```

3. 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, that is, [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.

Test cases:

#	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}

[10+15+10 + 5 =40]

```c
// Code Ceator: Arun B (arunbsmn@kgpian.iitkgp.ac.in)
#include <stdio.h>

// Function Prototypes
void createBins(int arr[], int size, int numBins, int minVal, int
maxVal);
int findMin(int arr[], int size);
int findMax(int arr[], int size);

int main()
{
    // Taking the size of the array and number of bins as input from
the user
    int size, numBins;

    printf("Enter the number of elements in the array: ");
    scanf("%d", &size);

    int arr[size];

    // Input the elements of the array
    printf("Enter the array elements:\n");
    for (int i = 0; i < size; i++)
    {
        scanf("%d", &arr[i]);
    }

    printf("Enter the number of bins: ");
    scanf("%d", &numBins);

    // Finding the minimum and maximum values in the array
    int minVal = findMin(arr, size);
    int maxVal = findMax(arr, size);

    // Create and display bins
    createBins(arr, size, numBins, minVal, maxVal);

    return 0;
}

// Function to find the minimum value in the array
int findMin(int arr[], int size)
{
    int min = arr[0];
    for (int i = 1; i < size; i++)
    {
        if (arr[i] < min)
        {
            min = arr[i];
        }
    }
    return min;
}

// Function to find the maximum value in the array
int findMax(int arr[], int size)
{
    int max = arr[0];
    for (int i = 1; i < size; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
        }
    }
    return max;
}

// Function to create bins and categorize array elements into bins
void createBins(int arr[], int size, int numBins, int minVal, int
maxVal)
{
    int binRange = (maxVal - minVal + 1) / numBins;

    // Create and initialize bins
    for (int bin = 0; bin < numBins; bin++)
    {
        int binStart = minVal + bin * binRange;
        int binEnd = (bin == numBins - 1) ? maxVal : (binStart +
binRange - 1);

        printf("Bin %d (%d - %d): ", bin + 1, binStart, binEnd);

        int count = 0; // to count the number of elements in the
current bin

        // Check which elements of the array fall in the current bin
        for (int i = 0; i < size; i++)
        {
            if (arr[i] >= binStart && arr[i] <= binEnd)
            {
                printf("%d ", arr[i]);
                count++;
            }
        }
        if (count == 0)
        {
            printf("No elements");
        }

        printf("\nNumber of elements in bin %d: %d\n\n", bin + 1,
count);
    }
}
```