Assignment - 11 (More on functions in C Language)

1. Write a function to calculate LCM of two numbers. (TSRS)

int findLCM(int n1, int n2)

{

int lcm, i = 2;

// if any one number is 0

if(n1 == 0 && n2 || n1 && n2 == 0)

{

printf("LCM is undefined.");

return -1;

}

else

{

// converting negative numbers (if any) into positive

if(n1 < 0)

n1 = n1 \* -1;

if(n2 < 0)

n2 = n2 \* -1;

// we will pick {greater number and its multiples} and will check the divisibility of each multiple of greater number with smaller number.

if(n1 > n2)

{

lcm = n1;

while(lcm % n2) // That smallest multiple of greater number (lcm) which is completely divisible by smaller number (n2) will be the LCM(n1, n2)

{

lcm = n1 \* i;

i++;

}

}

else

{

lcm = n2;

while(lcm % n1)

{

lcm = n2 \* i;

i++;

}

}

return lcm;

}

}

2. Write a function to calculate HCF of two numbers. (TSRS)

int findHCF(int n1, int n2)

{

int hcf = 1;

// negative numbers(if any) converted to positive

if(n1 < 0)

n1 = n1 \* -1;

if(n2 < 0)

n2 = n2 \* -1;

// HCF(0,0) is not defined

if(n1 == 0 && n2 == 0)

{

printf("HCF is not defined.");

return -1;

}

// if any one number is 0, then other non zero number is HCF

else if(n1 == 0 && n2)

return n2;

else if(n2 == 0 && n1)

return n1;

// if numbers are same, then print any one as hcf

else if(n1 == n2)

return n1;

// the smaller number is picked and all numbers from 2 to this number are divided by n1 and n2 to check divisibility.

// The last/highest possible factor of smaller number which also divides larger number will be HCF.

else if(n1 < n2)

{

for(int i = n1; i >= 2; i--)

{

if(n1 % i == 0 && n2 % i == 0)

{

hcf = i;

break;

}

}

return hcf;

}

else

{

for(int i = n2; i >= 2; i--)

{

if(n1 % i == 0 && n2 % i == 0)

{

hcf = i;

break;

}

}

return hcf;

}

}

3. Write a function to check whether a given number is Prime or not. (TSRS)

int isPrime(int number)

{

int i;

if(number <= 1)

return 0;

else

{

for(i = 2; i <= number / 2; i++)

{

if(number % i == 0)

break;

}

if(i <= number / 2)

return 0;

else

return 1;

}

}

4. Write a function to find the next prime number of a given number. (TSRS)

int isprime(int);

int isprime(int num)

{

if(num < 2)

return 0;

else

{

for(int i = 2; i <= num / 2; i++)

if(num % i == 0)

return 0;

return 1;

}

}

int next\_prime\_num(int);

int next\_prime\_num(int num)

{

int next\_num = num + 1;

while(1)

{

if(isprime(next\_num))

return next\_num;

else

next\_num++;

}

}

5. Write a function to print first N prime numbers (TSRN)

void printFirstNPrimeNumbers(int N)

{

int count, i, number = 2;

if(N <= 0)

printf("Invalid value of N entered.");

else

{

printf("First %d prime numbers are ", N);

while(N)

{

for(i = 2; i <= number / 2; i++)

{

if(number % i == 0)

break;

}

if(i > number / 2)

{

printf("%d, ", number);

N--;

}

number++;

}

printf("\b\b. ");

}

}

6. Write a function to print all Prime numbers between two given numbers. (TSRN)

int isprime(int);

int isprime(int num)

{

if(num < 2)

return 0;

else

{

for(int i = 2; i <= num / 2; i++)

if(num % i == 0)

return 0;

return 1;

}

}

void prime\_nums\_between\_2\_nums(int, int);

void prime\_nums\_between\_2\_nums(int n1, int n2)

{

printf("Prime Numbers between %d and %d are:\n", n1, n2);

for(int i = n1 + 1; i < n2; i++)

if(isprime(i))

printf("%d ", i);

}

7. Write a function to print first N terms of Fibonacci series (TSRN)

void printFirstNTermsOfFibonacciSeries(int N)

{

int f1 = 0, f2 = 1, i = 3, f3;

printf("%d %d ", f1, f2);

while(i <= N)

{

f3 = f1 + f2;

printf("%d ", f3);

f1 = f2;

f2 = f3;

i++;

}

}

8. Write a function to print PASCAL Triangle. (TSRN)

void printPascalTriangle(int N)

{

int arr[N][2 \* N - 1], i, j;

// initializing all the values of 2d array with 0

for(i = 0; i < N; i++)

{

for(j = 0; j < 2 \* N - 1; j++)

{

arr[i][j] = 0;

}

}

// building pascal triangle

for(i = 0; i < N; i++)

{

for(j = 0; j < 2 \* N - 1; j++)

{

if(j >= N - 1 - i && j <= N - 1 + i)

{

if(N % 2)

{

if(i % 2)

{

if(j % 2)

arr[i][j] = arr[i - 1][j - 1] + arr[i - 1][j + 1];

}

else if(i == 0)

{

arr[i][j] = 1;

}

else if(i == N - 1 && (j == 0 || j == 2 \* N - 2))

arr[i][j] = 1;

else

{

if(j % 2 == 0)

arr[i][j] = arr[i - 1][j - 1] + arr[i - 1][j + 1];

}

}

else

{

if(i % 2)

{

if(i == N - 1 && (j == 0 || j == 2 \* N - 2))

arr[i][j] = 1;

else if(j % 2 == 0)

arr[i][j] = arr[i - 1][j - 1] + arr[i - 1][j + 1];

}

else if(i == 0)

{

arr[i][j] = 1;

}

else

{

if(j % 2)

arr[i][j] = arr[i - 1][j - 1] + arr[i - 1][j + 1];

}

}

}

}

}

// printing pascal triangle

for(i = 0; i < N; i++)

{

for(j = 0; j < 2 \* N - 1; j++)

{

if(arr[i][j])

printf("%d", arr[i][j]);

printf("\t");

}

printf("\n\n");

}

}

9. Write a program in C to find the square of any number using the function.

#include<stdio.h>

int findSquare(int number)

{

return number \* number;

}

int main()

{

int number;

printf("Enter a number to find its square: ");

scanf("%d", &number);

printf("Square of %d is %d.", number, findSquare(number));

return 0;

}

10. Write a program in C to find the sum of the series 1! /1+2!/2+3!/3+4!/4+5!/5 using the function.

#include<stdio.h>

int factorial(int);

int seriesSum(int);

int factorial(int num)

{

int fact = 1;

if(num < 0)

return -1;

else if(num == 0)

return 1;

else

{

for(int i = 1; i <= num; i++)

fact = fact \* i;

return fact;

}

}

int seriesSum(int N)

{

int i, sum = 0;

for(i = 1; i <= N; i++)

{

sum = sum + factorial(i) / i;

}

return sum;

}

int main()

{

printf("Sum of series 1! /1+2!/2+3!/3+ ... + N! / N is %d.", seriesSum(5));

return 0;

}