

C Language Programs

Assignment 4

Submitted By:

Amitanshu Kumar Swain

1CSE 2Y Enrolment No.A2305220070

INDEX

|  |  |
| --- | --- |
| Experiment 1 | Write a C program to print all-natural numbers between 1 to n using function. |
| Experiment 2 | Write a C program to print all even or odd numbers in given range using recursion. |
| Experiment 3 | Write a C program to find sum of all-natural numbers between 1 to n using recursion. |
| Experiment 4 | Write a C program to find sum of all even or odd numbers in given range using recursion. |
| Experiment 5 | Write a C program to find reverse of any number using recursion. |
| Experiment 6 | Write a C program to check whether a number is palindrome or not using recursion. |
| Experiment 7 | Write a C program to find sum of digits of a given number using recursion. |
| Experiment 8 | Write a C program to generate nth Fibonacci term using recursion. |
| Experiment 9 | Write a C program to find GCD (HCF) of two numbers using recursion. |
| Experiment 10 | Write a C program to find LCM of two numbers using recursion function. |

Experiment 1

Objective: Write a C program to print all-natural numbers between 1 to n using function.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

#include <stdio.h>

void printNaturalNumbers(int lowerLimit, int upperLimit);

int main()

{

int lowerLimit, upperLimit;

printf("Enter lower limit: ");

scanf("%d", &lowerLimit);

printf("Enter upper limit: ");

scanf("%d", &upperLimit);

printf("All natural numbers from %d to %d are: ", lowerLimit, upperLimit);

printNaturalNumbers(lowerLimit, upperLimit);

return 0;

}

void printNaturalNumbers(int lowerLimit, int upperLimit)

{

if(lowerLimit > upperLimit)

return;

printf("%d, ", lowerLimit);

printNaturalNumbers(lowerLimit + 1, upperLimit);

}

Output:

Enter lower limit:

Enter upper limit:

All numbers from (say 1 to 100) are:

Experiment 2

Objective: Write a C program to print all even or odd numbers in given range using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**void printEvenOdd(int cur, int limit);**

**int main()**

**{**

**int lowerLimit, upperLimit;**

**printf("Enter lower limit: ");**

**scanf("%d", &lowerLimit);**

**printf("Enter upper limit: ");**

**scanf("%d", &upperLimit);**

**printf("Even/odd Numbers from %d to %d are: ", lowerLimit, upperLimit);**

**printEvenOdd(lowerLimit, upperLimit);**

**return 0;**

**}**

**void printEvenOdd(int cur, int limit)**

**{**

**if(cur > limit)**

**return;**

**printf("%d, ", cur);**

**printEvenOdd(cur + 2, limit);**

**}**

Output:

Enter lower limit:

Enter upper limit:

All odd numbers between (say 1 to 100) are:

Experiment 3

Objective: Write a C program to find sum of all-natural numbers between 1 to n using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**int sumOfNaturalNumbers(int start, int end);**

**int main()**

**{**

**int start, end, sum;**

**printf("Enter lower limit: ");**

**scanf("%d", &start);**

**printf("Enter upper limit: ");**

**scanf("%d", &end);**

**sum = sumOfNaturalNumbers(start, end);**

**printf("Sum of natural numbers from %d to %d = %d", start, end, sum);**

**return 0;**

**}**

**int sumOfNaturalNumbers(int start, int end)**

**{**

**if(start == end)**

**return start;**

**else**

**return start + sumOfNaturalNumbers(start + 1, end);**

**}**

Output:

Enter lower limit:

Enter upper limit:

Sum of natural numbers from (say 1 to 100) is:

Experiment 4

Objective: Write a C program to find sum of all even or odd numbers in given range using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**int sumOfEvenOdd(int start, int end);**

**int main()**

**{**

**int start, end, sum;**

**printf("Enter lower limit: ");**

**scanf("%d", &start);**

**printf("Enter upper limit: ");**

**scanf("%d", &end);**

**printf("Sum of odd numbers between %d to %d = %d\n", start, end, sumOfEvenOdd(start, end));**

**return 0;**

**}**

**int sumOfEvenOdd(int start, int end)**

**{**

**if(start > end)**

**return 0;**

**else**

**return (start + sumOfEvenOdd(start + 2, end));**

**}**

Output:

Enter lower limit:

Enter upper limit:

Sum of odd numbers between (say 1 to 100) =

Experiment 5

Objective: Write a C program to find reverse of any number using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**#include <math.h>**

**int reverse(int num);**

**int main()**

**{**

**int num, rev;**

**printf("Enter any number: ");**

**scanf("%d", &num);**

**rev = reverse(num);**

**printf("Reverse of %d = %d", num, rev);**

**return 0;**

**}**

**int reverse(int num)**

**{**

**int digit = (int) log10(num);**

**if(num == 0)**

**return 0;**

**return ((num%10 \* pow(10, digit)) + reverse(num/10));**

**}**

Output:

Enter any number:

Reverse:

Experiment 6

Objective: Write a C program to check whether a number is palindrome or not using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**#include <math.h>**

**int reverse(int num);**

**int isPalindrome(int num);**

**int main()**

**{**

**int num;**

**printf("Enter any number: ");**

**scanf("%d", &num);**

**if(isPalindrome(num) == 1)**

**{**

**printf("%d is palindrome number.\n", num);**

**}**

**else**

**{**

**printf("%d is NOT palindrome number.\n", num);**

**}**

**return 0;**

**}**

**int isPalindrome(int num)**

**{**

**if(num == reverse(num))**

**{**

**return 1;**

**}**

**return 0;}**

**int reverse(int num)**

**{**

**int digit = (int)log10(num);**

**if(num == 0)**

**return 0;**

**return ((num%10 \* pow(10, digit)) + reverse(num/10));**

**}**

Output:

Enter any number:

\_\_\_\_ is not/a palindrome.

Experiment 7

Objective: Write a C program to find sum of digits of a given number using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**int sumOfDigits(int num);**

**int main()**

**{**

**int num, sum;**

**printf("Enter any number to find sum of digits: ");**

**scanf("%d", &num);**

**sum = sumOfDigits(num);**

**printf("Sum of digits of %d = %d", num, sum);**

**return 0;**

**}**

**int sumOfDigits(int num)**

**{**

**if(num == 0)**

**return 0;**

**return ((num % 10) + sumOfDigits(num / 10));**

**}**

Output:

Enter any number to find sum of digits:

Sum of digits=

Experiment 8

Objective: Write a C program to generate nth Fibonacci term using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**unsigned long long fibo(int num);**

**int main()**

**{**

**int num;**

**unsigned long long fibonacci;**

**printf("Enter any number to find nth fiboacci term: ");**

**scanf("%d", &num);**

**fibonacci = fibo(num);**

**printf("%d fibonacci term is %llu", num, fibonacci);**

**return 0;**

**}**

**unsigned long long fibo(int num)**

**{**

**if(num == 0)**

**return 0;**

**else if(num == 1)**

**return 1;**

**else**

**return fibo(num-1) + fibo(num-2);**

**}**

Output:

Enter any number to find the nth Fibonacci term:

\_\_\_ Fibonacci term is

Experiment 9

Objective: Write a C program to find GCD (HCF) of two numbers using recursion.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**int gcd(int a, int b);**

**int main()**

**{**

**int num1, num2, hcf;**

**printf("Enter any two numbers to find GCD: ");**

**scanf("%d%d", &num1, &num2);**

**hcf = gcd(num1, num2);**

**printf("GCD of %d and %d = %d", num1, num2, hcf);**

**return 0;**

**}**

**int gcd(int a, int b)**

**{**

**if(b == 0)**

**return a;**

**else**

**return gcd(b, a%b);**

**}**

Output:

Enter any two numbers to find GCD:

GCD of \_\_ and \_\_ =

Experiment 10

Objective: Write a C program to find LCM of two numbers using recursion function.

Language Used: C

Theory: The compiler is a type of language processor which converts a high level program into machine level program at one go rather than line by line. The language processor converts all user instructions into machine understandable language.

Program:

**#include <stdio.h>**

**int lcm(int a, int b);**

**int main()**

**{**

**int num1, num2, LCM;**

**printf("Enter any two numbers to find lcm: ");**

**scanf("%d%d", &num1, &num2);**

**if(num1 > num2)**

**LCM = lcm(num2, num1);**

**else**

**LCM = lcm(num1, num2);**

**printf("LCM of %d and %d = %d", num1, num2, LCM);**

**return 0;**

**}**

**int lcm(int a, int b)**

**{**

**static int multiple = 0;**

**multiple += b;**

**if((multiple % a == 0) && (multiple % b == 0))**

**{**

**return multiple;**

**}**

**else**

**{**

**return lcm(a, b);**

**}**

**}**

Output:

Enter any two numbers to find the LCM:

LCM of \_ and \_ =