Assignment - 13 (More on Recursion in C Language)

1. Write a recursive function to calculate sum of first N natural numbers

int sumOfFirstNNaturalNums(int);

int sumOfFirstNNaturalNums(int n)

{

if(n == 1)

return 1;

return n + sumOfFirstNNaturalNums(n - 1);

}

2. Write a recursive function to calculate sum of first N odd natural numbers

int sumOfFirstNOddNaturalNums(int);

int sumOfFirstNOddNaturalNums(int n)

{

if(n == 1)

return 1;

return 2 \* n - 1 + sumOfFirstNOddNaturalNums(n - 1);

}

3. Write a recursive function to calculate sum of first N odd natural numbers

int sumOfFirstNOddNaturalNums(int);

int sumOfFirstNOddNaturalNums(int n)

{

if(n == 1)

return 1;

return 2 \* n - 1 + sumOfFirstNOddNaturalNums(n - 1);

}

4. Write a recursive function to calculate sum of squares of first n natural numbers

int sumOfSquaresOffirstNNaturalNums(int);

int sumOfSquaresOffirstNNaturalNums(int n)

{

if(n == 1)

return 1;

return n \* n + sumOfSquaresOffirstNNaturalNums(n - 1);

}

5. Write a recursive function to calculate sum of digits of a given number

int sumOfDigitsOfNum(int n)

{

if(n == 0)

return 0;

return n % 10 + sumOfDigitsOfNum(n / 10);

}

6. Write a recursive function to calculate factorial of a given number

int factorial(int);

int factorial(int number)

{

if(number == 1 || number == 0)

return 1;

return number \* factorial(number - 1);

}

7. Write a recursive function to calculate HCF of two numbers

int smallnum(int a, int b)

{

if(a < b)

return a;

else

return b;

}

int largenum(int a, int b)

{

if(a > b)

return a;

else

return b;

}

int hcf(int n1, int n2)

{

// 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.\n");

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;

else

{

int s = smallnum(n1, n2), g = largenum(n1, n2);

int rem = g % s;

if(rem == 0)

return s;

return hcf(s, rem);

}

}

8. Write a recursive function to print first N terms of Fibonacci series

#include<stdio.h>

int nthFibonacciTerm(int n)

{

if(n <= 0)

return 0;

else if(n == 1)

return 0;

else if(n == 2)

return 1;

else

return nthFibonacciTerm(n - 1) + nthFibonacciTerm(n - 2);

}

void printFirstNTermsOfFibonacciSeries(int N)

{

if(N == 1)

printf("0 ");

else

{

printFirstNTermsOfFibonacciSeries(N - 1);

printf("%d ", nthFibonacciTerm(N));

}

}

int main()

{

printFirstNTermsOfFibonacciSeries(10);

return 0;

}

9. Write a program in C to count the digits of a given number using recursion.

#include<stdio.h>

int recursiveDigitsCount(int number, int count)

{

if(number)

{

count++;

recursiveDigitsCount(number / 10, count);

}

else

return count;

}

int digitsCount(int number)

{

if(number == 0)

return 1;

else

return recursiveDigitsCount(number, 0);

}

int main()

{

int number;

printf("Enter a number to find number of digits of this number: ");

scanf("%d", &number);

printf("Number of digits in %d is %d.", number, digitsCount(number));

return 0;

}

10. Write a program in C to calculate the power of any number using recursion.

#include<stdio.h>

double baseRaisedToPower(double base, int power)

{

if(base < 0)

{

if(power % 2)

return -1 \* baseRaisedToPower(base \* - 1, power);

else

return baseRaisedToPower(base \* -1, power);

}

if(base == 0)

return 0;

else if(power == 0)

return 1;

else

return base \* baseRaisedToPower(base, power - 1);

}

int main()

{

int power;

double base;

printf("Enter the base and power to find base raised to power: ");

scanf("%lf %d", &base, &power);

printf("%lf raised to %d is %lf.", base, power, baseRaisedToPower(base, power));

return 0;

}