Assignment - 7 (Iterative Control Statements (Part - 2))

1. Write a program to find the Nth term of the Fibonnaci series.

#include<stdio.h>

int main()

{

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

printf("Enter a number: ");

scanf("%d", &N);

if(N == 1)

printf("1st term of fibonacci series is 0.");

else if(N == 2)

printf("2nd Term of fibonacci series is 1.");

else

{

while(i <= N)

{

f3 = f1 + f2;

f1 = f2;

f2 = f3;

i++;

}

printf("%dth term of fibonacci series is %d.", N, f3);

}

return 0;

}

2. Write a program to print first N terms of Fibonacci series

#include<stdio.h>

int main()

{

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

printf("Enter value of N: ");

scanf("%d", &N);

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

while(i <= N)

{

f3 = f1 + f2;

printf("%d ", f3);

f1 = f2;

f2 = f3;

i++;

}

return 0;

}

3. Write a program to check whether a given number is there in the Fibonacci

series or not.

#include<stdio.h>

int main()

{

int number, f1 = 0, f2 = 1, f3 = -1;

printf("Enter a number: ");

scanf("%d", &number);

if(number < 0)

printf("%d is not a fibonacci number.", number);

else if(number == 0)

printf("%d is a fibonacci number.", number);

else

{

while(f3 < number)

{

f3 = f1 + f2;

f1 = f2;

f2 = f3;

}

if(f3 == number)

printf("%d is a fibonacci number.", number);

else

printf("%d is not a fibonacci number.", number);

}

return 0;

}

4. Write a program to calculate HCF of two numbers

#include<stdio.h>

int main()

{

int n1, n2, hcf = 1;

printf("Enter two numbers to find their HCF: ");

scanf("%d%d", &n1, &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.");

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

else if(n1 == 0 && n2)

printf("HCF is %d.", n2);

else if(n2 == 0 && n1)

printf("HCF is %d.", n1);

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

else if(n1 == n2)

printf("HCF is %d.", 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;

}

}

printf("HCF is %d.", hcf);

}

else

{

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

{

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

{

hcf = i;

break;

}

}

printf("HCF is %d.", hcf);

}

return 0;

}

5. Write a program to check whether two given numbers are co-prime

numbers or not

#include<stdio.h>

int main()

{

int n1, n2;

printf("Enter two numbers to check whether they are co-prime or not: ");

scanf("%d%d", &n1, &n2);

if(n1 < 0)

n1 = n1 \* -1;

if(n2 < 0)

n2 = n2 \* -1;

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

printf("Numbers are not co prime.");

else if(n1 == 1 || n2 == 1)

printf("Numbers are co prime.");

else

{

int small\_num, count = 0;

if(n1 < n2)

small\_num = n1;

else

small\_num = n2;

for(int i = 2; i <= small\_num; i++)

{

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

{

count++;

break;

}

}

if(count)

printf("Numbers are not co prime.");

else

printf("Numbers are co prime.");

}

return 0;

}

6. Write a program to print all Prime numbers under 100

#include<stdio.h>

int main()

{

int count;

printf("All prime numbers under 100 are:\n");

for(int i = 2; i < 100; i++)

{

count = 0;

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

{

if(i % j == 0)

{

count++;

break;

}

}

if(count == 0)

printf("%d ", i);

}

return 0;

}

7. Write a program to print all Prime numbers between two given numbers

#include<stdio.h>

int main()

{

int count, num1, num2;

printf("Enter two numbers between which all prime numbers are to be printed: ");

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

for(int i = num1 + 1; i < num2; i++)

{

count = 0;

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

{

if(i % j == 0)

{

count++;

break;

}

}

if(count == 0)

printf("%d ", i);

}

return 0;

}

8. Write a program to find next Prime number of a given number

#include<stdio.h>

int main()

{

int number;

printf("Enter a number to print next prime number after this number: ");

scanf("%d", &number);

if(number <= 1)

printf("Next prime number after %d is 2.", number);

else

{

int next\_prime = number, count = -1;

while(count)

{

count = 0;

next\_prime++;

for(int i = 2; i <= next\_prime / 2; i++)

{

if(next\_prime % i == 0)

{

count++;

break;

}

}

}

printf("Next prime number after %d is %d.",number, next\_prime);

}

return 0;

}

9. Write a program to check whether a given number is an Armstrong number

or not

#include<stdio.h>

int digits\_count(int);

int digits\_count(int number)

{

int count = 0;

while(number)

{

count++;

number = number / 10;

}

return count;

}

// function to find powers of int type base rased to int type positive Exponent

int baseToExp(int, int);

int baseToExp(int base, int exponent)

{

if(exponent < 0)

return -1;

else if(exponent == 0)

return 1;

else

{

int pro = 1;

while(exponent)

{

pro = pro \* base;

exponent--;

}

return pro;

}

}

int isarmstrong(int);

int isarmstrong(int num)

{

if(num < 0)

return 0;

else

{

int x = num, sum = 0, digit;

while(x)

{

digit = x % 10;

sum = sum + baseToExp(digit, digits\_count(num));

x = x / 10;

}

if(sum == num)

return 1;

else

return 0;

}

}

int main()

{

int number;

printf("Enter a number to check whether the number is an armstrong number or not: ");

scanf("%d", &number);

if(isarmstrong(number))

printf("%d is an armstrong number.", number);

else

printf("%d is not an armstrong number.", number);

return 0;

}

10. Write a program to print all Armstrong numbers under 1000

#include<stdio.h>

int digits\_count(int);

int digits\_count(int number)

{

int count = 0;

while(number)

{

count++;

number = number / 10;

}

return count;

}

// function to find powers of int type base rased to int type positive Exponent

int baseToExp(int, int);

int baseToExp(int base, int exponent)

{

if(exponent < 0)

return -1;

else if(exponent == 0)

return 1;

else

{

int pro = 1;

while(exponent)

{

pro = pro \* base;

exponent--;

}

return pro;

}

}

int isarmstrong(int);

int isarmstrong(int num)

{

if(num < 0)

return 0;

else

{

int x = num, sum = 0, digit;

while(x)

{

digit = x % 10;

sum = sum + baseToExp(digit, digits\_count(num));

x = x / 10;

}

if(sum == num)

return 1;

else

return 0;

}

}

void printArmstrongNumsUnder1000()

{

printf("All armstrong numbers under 1000 are:\n");

for(int i = 0; i < 1000; i++)

if(isarmstrong(i))

printf("%d ", i);

}

int main()

{

printArmstrongNumsUnder1000();

return 0;

}