**Project-2**

**Name: Pawan Dhungana**

**Project no: 2**

**Due Date: September 13, 2018**

**Design Document**

**Introduction**

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

**Data Structures**

In this program, recursion is used where the function takes in a positive integer value and the value is passed into a function which returns the integer in reverse order.

**Functions**

There is 1 function in the this program: backwards() that takes in a parameter and prints the integers in the screen in reverse order.

**Main Program**

In the main program, the recursive function is being called.

In the main program, the functions are being called to read the file and check if the integer is present in the array.

**User Document**

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

The program's name is Project2.cpp, to compile and run it, simply enter:

g++ Project2.cpp

a.out

A run of the program might look like this:

Enter the integer:

123

The number printed backwards is: 321

**Code Listing:**

//Pawan Dhungana

//CS-301 Project 2

//Print integer backwards

//Due date: 09/13/2018

#include<iostream>

#include<cstdlib>

using namespace std;

int backwards(int);

int main()

{

int number;

int x=2;

cout<<"\nEnter the integer : "<<endl;

cin>>number; //takes in the integer

cout<<"The number printed backwards is: ";

cout<<backwards(number);

}

int backwards(int a)

{

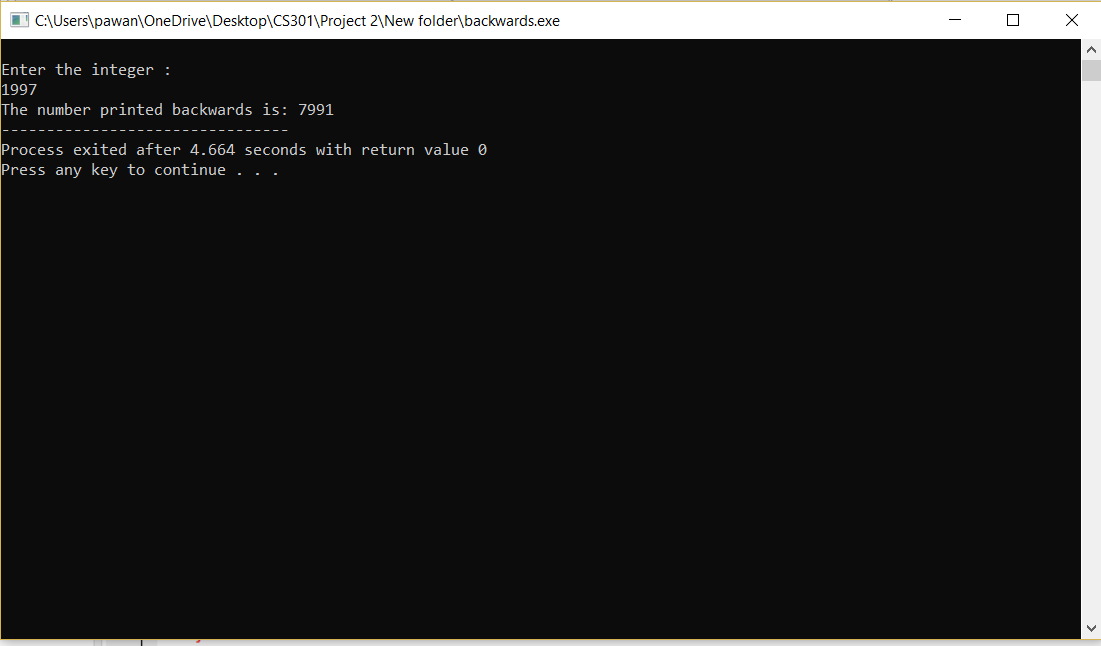
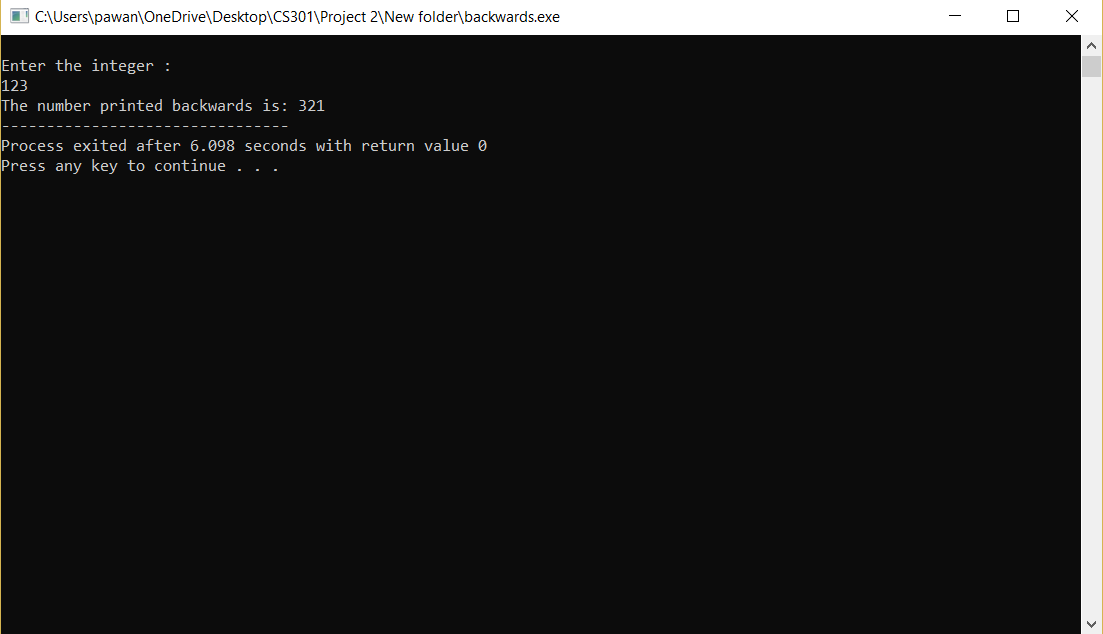
if(a < 10) //base/basic case i.e if number is less than 10 the problem is already solved, nothing to do, so simply return

return a;

else{

cout << a % 10; // print that last digit, e.g 103%10 == 3

return backwards(a/10); //solve the same problem but with smaller number, i.e make the problem smaller by dividing it by 10, initially we had 103, now 10

**Test Document**

**Design Document**

**Introduction**

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

**Data Structures**

In this program, a file is opened, and the integers are read and stored in an array. A character is introduced to take in the file. Then recursion is used to check if the integer entered by the user is present in the array.

**Functions**

In this program, there are two functions:

Readfile(): a void function that read the file and store the numbers in the file into an array

Present(): an int function that returns whether the integer entered by the user is present in the array.

**Main Program**

In the main program, two functions are being called to read the file and check if the integer is present in the array.

**User Document**

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

The program's name is Project2.cpp, to compile and run it, simply enter:

g++ Project2.cpp

a.out

A run of the program might look like this:

Data file name -> array.dat

The array read from the file is: 1 2 4 89 65 78 15

Test value -> 2

Present

Test value -> 99

Not Present

**Code Listing:**

//Pawan Dhungana

//CS-301 Project 2

//Check if the number is present in an array

//Due date: 09/13/2018

#include<iostream>

#include<fstream>

#include<cstdlib>

using namespace std;

void readfile(int a[],int n, int k);

int present(int a[],int n,int k);

int main()

{

int n=7; //size of array

int a[n]; //array declaration

int k; //the number that user enter

readfile(a,n,k);

cout<<"\nTest value -> ";

cin>>k;

if ( present(a,n,k) != false)

cout<<"\nPresent"<<endl;

else

cout<<"\nNot Present";

}

void readfile(int a[],int n, int k)

{

cout<<"Data file name -> ";

ifstream in\_f;

char file[50];//takes input what file user want to open

cin.getline(file,50);

in\_f.open(file);//opens file

cout<<"\nThe array read from the file is : ";

for(int i=0;i<n;i++) //stores the numbers read from file into an array

{

in\_f >> a[i];

cout<<a[i]<<" ";

}

in\_f.close(); //closes the file

}

int present(int a[],int n,int k) //check if the number entered is present in the array

{

if(n<0) //base case

return false;

else if(a[n]==k) // if the element is present

return true;

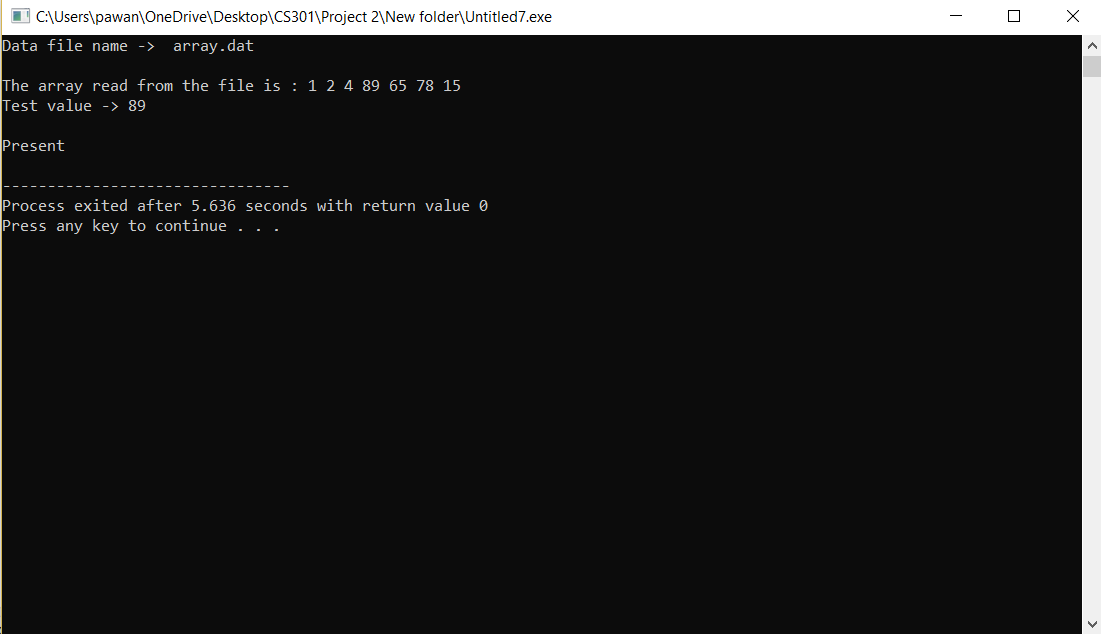
else //recursion

return present(a,n-1,k);

}

**Test Document**





**Design Document**

**Introduction**

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

**Data Structures**

In this program, recursion is used where the function takes in a positive integer value and the value is passed into a function which returns the prime factors of the integer.

**Functions**

In this program, there is 1 function: prime() that takes in two parameters and prints the prime factors of the integer.

**Main Program**

In the main program, the recursive function is being called.

**User Document**

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

The program's name is Project2.cpp, to compile and run it, simply enter:

g++ Project2.cpp

a.out

A run of the program might look like this:

Enter a positive integer: 5432

The prime factors of 5432 are: 2 2 2 7 97

**Code Listing:**

//Pawan Dhungana

//CS-301 Project 2

//Prime factorization

//Due date: 09/13/2018

#include<iostream>

#include<cstdlib>

using namespace std;

int prime(int,int);

int main()

{

int number;

int x=2;

cout<<"\nEnter the integer : "<<endl;

cin>>number;

cout<<"The prime factors are: ";

cout<<prime(number,x);

}

int prime(int num,int div)

{

if(num == 1) //base case if number is equal to 1, return 1

return num;

else if(num%div==0) //check if the integer is divisible and print divisor

{

cout<<div<<" ";

return prime(num/div,div);

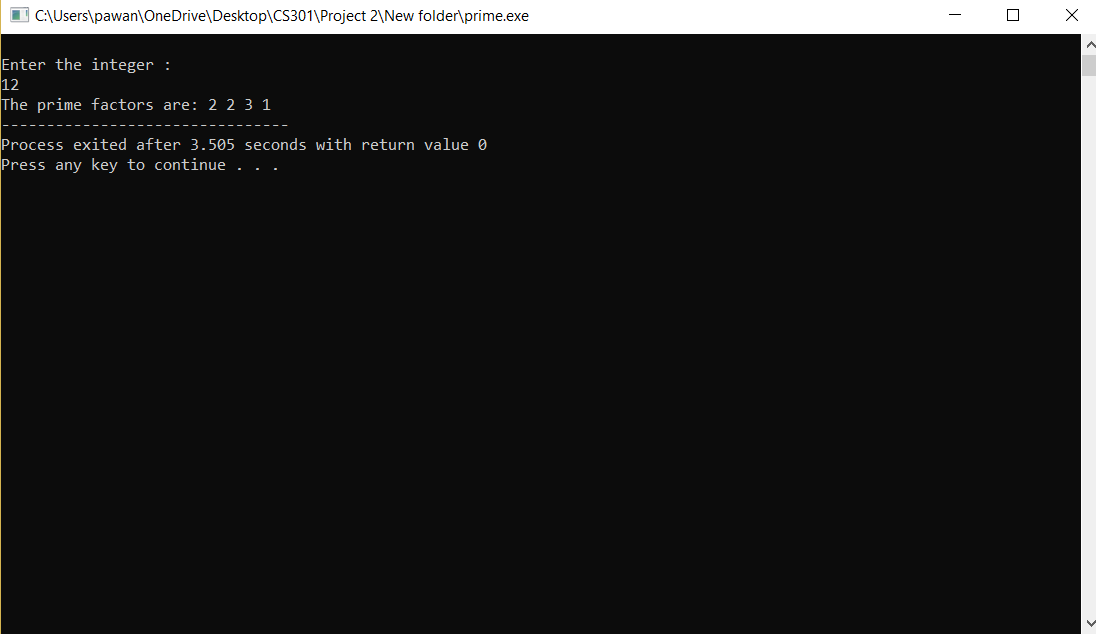
}

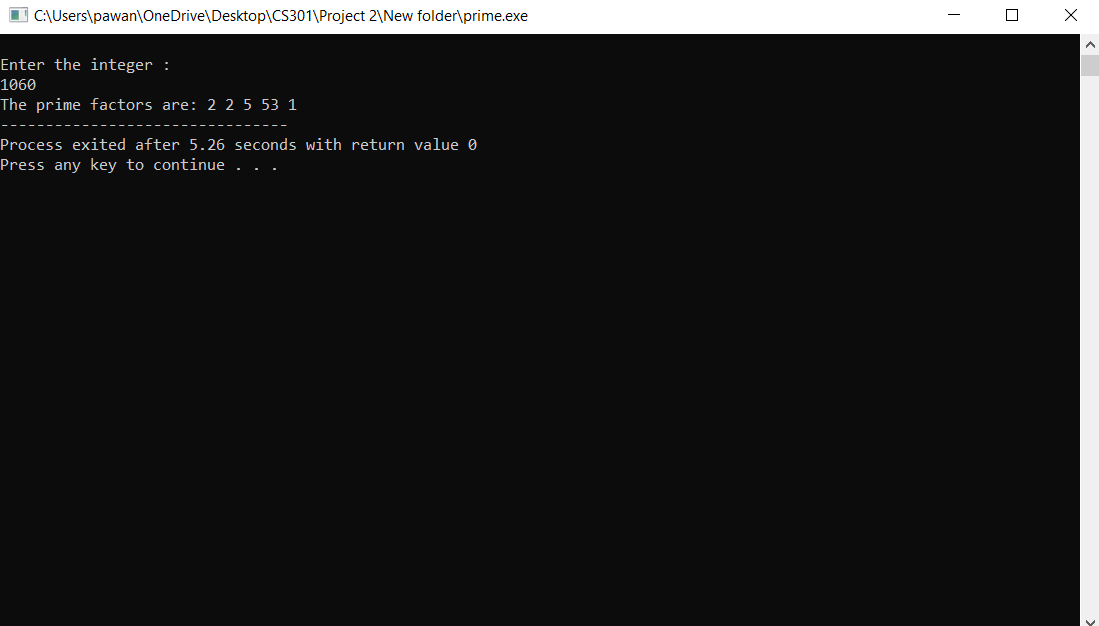
else

return prime(num,div+1); //increase the dividend and check if now divisible

}

**Test Document**

****

****

**Summary**

In this project, we implemented recursion in three different programs. Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

From this project we learned that recursion can be done as an alternative of iteration. It is generally shorter in code than that of iterative version, but the effectiveness depends upon the problem itself. In this project we asked the user for an integer and printed it backwards. We, also, opened a file of numbers, read the file and stored it into an array and checked if the user entered number is present in the array. Finally, we asked the user for an integer value and printed its prime factors.