CSC26 : Lab-III (OOP)

Assignment V

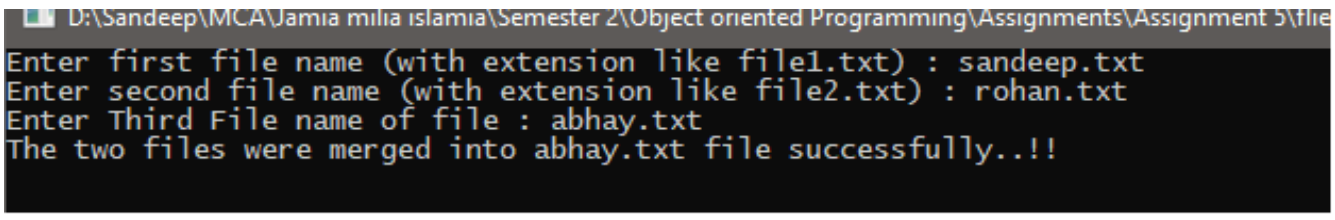
Sandeep Bhatt ,MCA051

Q1**: Write a program to merge the contents of two given flies into a third file.**

```
#include<iostream>
#include<conio.h>
#include<fstream>
#include<stdio.h>
#include<stdlib.h>
using namespace std;
int main()
{
 ifstream ifiles1, ifiles2;
 ofstream ifilet;
 char ch, fname1[20], fname2[20], fname3[30];
 cout<<"Enter first file name (with extension like file1.txt) : ";
 gets(fname1);
 cout<<"Enter second file name (with extension like file2.txt) : ";
 gets(fname2);
 cout<<"Enter Third File name of file : ";
 gets(fname3);
 ifiles1.open(fname1);
 ifiles2.open(fname2);
 if(ifiles1==NULL || ifiles2==NULL)
 {
  perror("Error Message ");
  cout<<"Press any key to exit...\n";
  getch();
  exit(EXIT_FAILURE);
 }
 ifilet.open(fname3);
```

```
if(!ifilet)
{
 perror("Error Message ");
 cout<<"Press any key to exit...\n";
 getch();
 exit(EXIT_FAILURE);
}
while(ifiles1.eof()==0)
{
 ifiles1>>ch;
 ifilet<<ch;
}
while(ifiles2.eof()==0)
{
 ifiles2>>ch;
 ifilet<<ch;
}
cout<<"The two files were merged into "<<fname3<<" file successfully..!!";
ifiles1.close();
ifiles2.close();
ifilet.close();
getch();
}
```

Output:

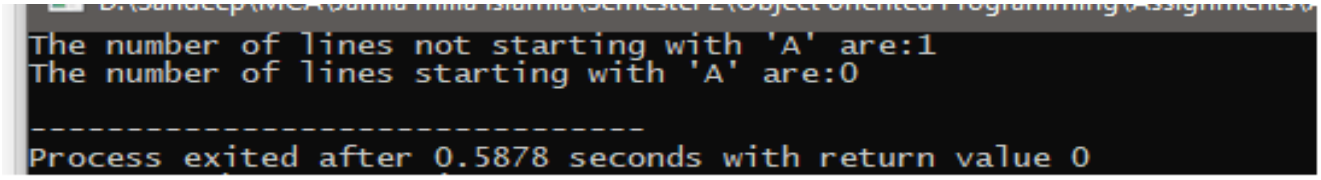


2. Write a function in C++ to count and display the number of lines not starting with alphabet 'A' present in a text file "Sandeep.TXT".

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
   ifstream fin;
   fin.open("sandeep.txt");

   char str[100];
   int count=0;
   int count1=0;

   while(!fin.eof())
   {
      fin.getline(str,100);
      if(str[0]!='A')
      {
         count++;
      }
      else if(str[0]=='A'&&str[1]==' ')
      {
         count1++;
      }
   }
```

```
    cout<<"The number of lines not starting with 'A' are:"<<count<<"\n";
    cout<<"The number of lines starting with 'A' are:"<<count1<<"\n";

    fin.close();
    return 0;
}
```

OUTPUT



3. Write a program using generic stack class to implement all possible stack operations using pointers.

```
#include<iostream>
#include<stdlib.h>
#include<stdio.h>
using namespace std;

template<class T>
class STACK{
    T* arr;
    int tos, length;
public:
STACK(int n)
{
    length = n;
```

```
    arr = new T[length];
    tos = -1;
}

void push(T item)
{
  arr[++tos] = item;
  cout << arr[tos] << " is Pushed into stack.";
}

void pop()
{
  cout << arr[tos--] << " is poped from stack.";
}

T top()
{
    return (arr[tos]);
}
bool isfull()
{
    if(tos == length - 1)
      return true;
    else
      return false;
}
bool isempty()
```

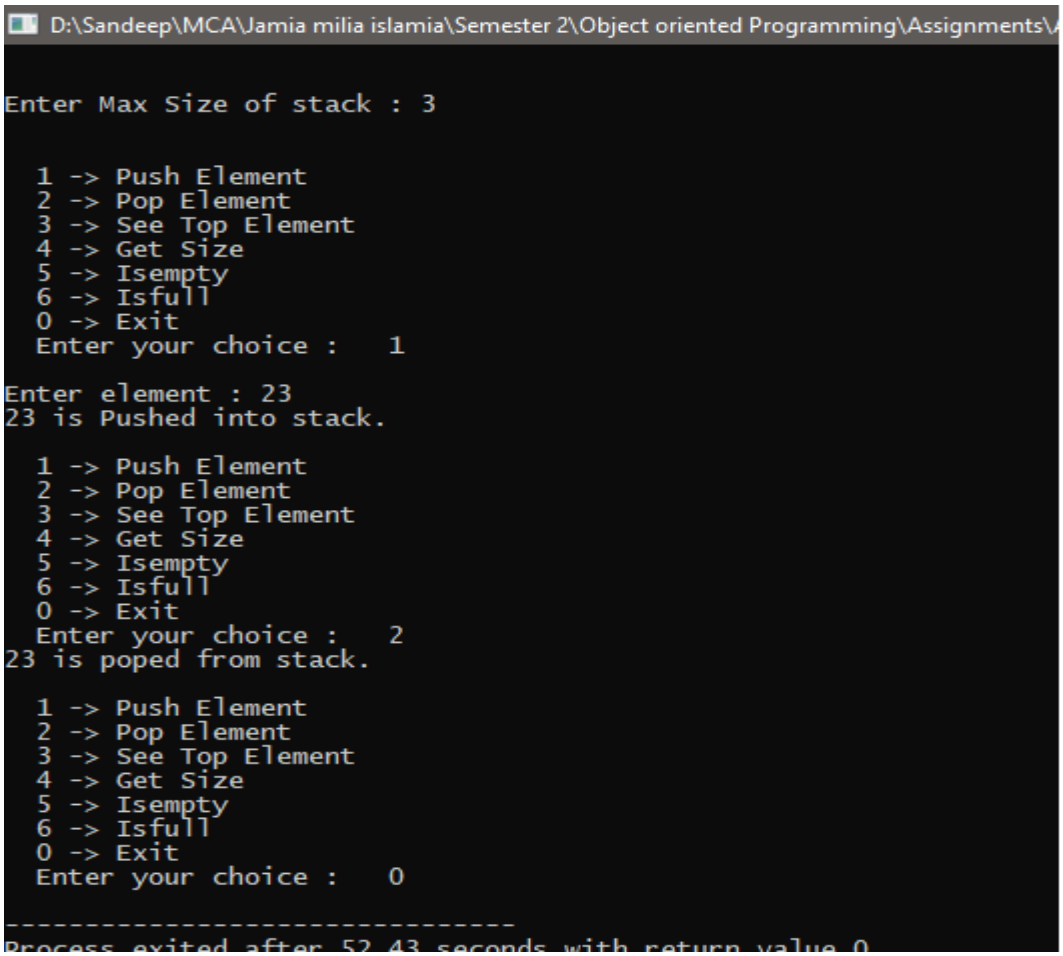
```
{
   if(tos == -1)
    return true;
   else
    return false;
}
int size()
{
   return tos + 1;
}
};
int main()
{
   int n, choice;

   do{
     cout << endl << endl << "Enter Max Size of stack : ";
     cin >> n;

     if(n < 1) cout <<"Invalid input...! Try Again.";
    }while(n < 1);

   STACK <int> s(n);
   int intItem;

   //STACK <char> s(n);     In case you have to create stack of  type char
   //char charItem;
```

```
while(1)
{
  cout << endl << endl << "  1 -> Push Element";
  cout << endl << "  2 -> Pop Element";
  cout << endl << "  3 -> See Top Element";
  cout << endl << "  4 -> Get Size";
  cout << endl << "  5 -> Isempty";
  cout << endl << "  6 -> Isfull";
  cout << endl << "  0 -> Exit";

  cout << endl << "  Enter your choice : \t";
  cin >> choice;

  switch(choice)
  {
    case 1: if(s.isfull())
              cout << endl << "Stack Overflow...!";
            else
            {
              cout << endl << "Enter element : ";
              cin >> intItem;
              s.push(intItem);
            }
            break;
```

```
case 2: if(s.isempty())
        cout << endl << "Stack Underflow...!";
       else
       {
        s.pop();
       }
       break;

case 3: if(s.isempty())
        cout << endl << "Stack Underflow...!";
       else
       {
        cout << s.top() << " is Top Element.";
       }
       break;

case 4: cout << endl << "Size of stack is : " << s.size();
       break;

case 5: if(s.isempty())
        cout << endl << "Yes, Stack is Empty!";
       else
        cout << endl << "No, Stack is Not Empty!";
       break;

case 6: if(s.isfull())
        cout << endl << "Yes, Stack is Full!";
```

```
            else
             cout << endl << "No, Stack is Not Full!";
            break;

     case 0: exit(EXIT_SUCCESS);
            break;
     default: cout << endl << "Invalid input...! Try Again.";
    }
  }
  return 0;
}
```

Output:



4. Write a program of your choice to handle the occurring exceptions in the program using multiple catch statements.

```
#include <iostream>
using namespace std;

class Test {
public:
	Test() { cout << "Constructor of Test " << endl; }
	~Test() { cout << "Destructor of Test " << endl; }
};

int main()
{
	try {
		Test t1;
		throw 14;
	}
	catch (int i) {
		cout << "Caught " << i << endl;
	}
}
```

Output: