Name: **Muhammad Anas Baig**

Enrollment No.: **01-134152-037**

Section: **BS(CS)-4A**

LAB-JOURNAL-2

Exercise 1:

Write a C++ program that prompts user to enter a number (in decimal). Convert the number into binary and display the binary number using the Stack.

Solution:

stack.h File:

```
#pragma once
class stack
{
private:
    int *arrStack;
    int top;
    int size;
public:
    stack(void);
    stack(int);
    bool isEmpty();
    bool isFull();
    void push(int);
    int pop();
    int _top();
    void display();
};
```

stack.cpp File:

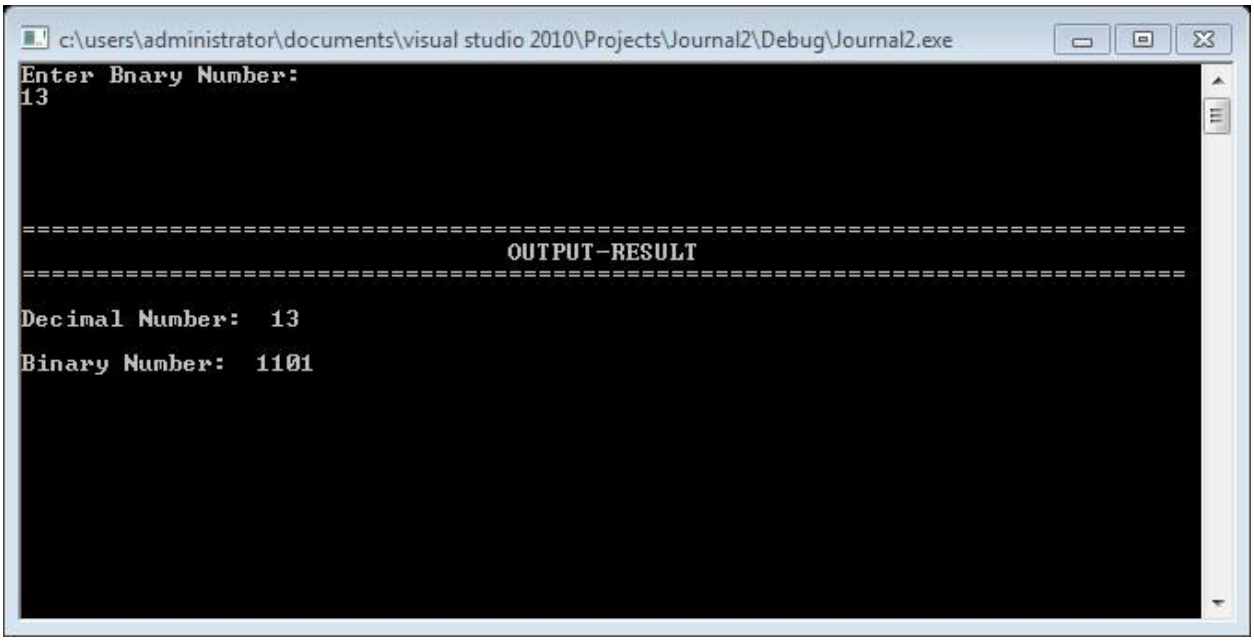
```
#include "stack.h"
#include <iostream>
using namespace std;

stack::stack(void)
{
    size = 100;
    arrStack = new int [size];
    top = -1;
}

stack::stack(int size)
{
    this->size = size;
    arrStack = new int [this->size];
    top = -1;
}

bool stack::isEmpty()
{
```

```
21.     if(top == -1)
22.     {
23.         return true;
24.     }
25.     else
26.     {
27.         return false;
28.     }
29. }
30.
31. bool stack::isFull()
32. {
33.     if(top == (size-1))
34.     {
35.         return true;
36.     }
37.     else
38.     {
39.         return false;
40.     }
41. }
42.
43. void stack::push(int value)
44. {
45.     if(!isFull())
46.     {
47.         arrStack[++top] = value;
48.     }
49.     else
50.     {
51.         cout<<"Stack Overflow!!!"<<endl;
52.     }
53. }
54.
55. int stack::pop()
56. {
57.     if(!isEmpty())
58.     {
59.         return (arrStack[top--]);
60.     }
61.     else
62.     {
63.         cout<<"Stack Underflow!!!"<<endl;
64.     }
65. }
66.
67. int stack::_top()
68. {
69.     if(!isEmpty())
70.     {
71.         return arrStack[top];
72.     }
73.     else
74.     {
75.         cout<<"Stack Empty!!!";
76.     }
77. }
78.
79. void stack::display()
80. {
81.     for(int i=top; i>=0; i--)
82.     {
83.         cout<<arrStack[i];
84.     }
85. }
```

main.cpp File:

```
1.  #include "stack.h"
2.  #include "conio.h"
3.  #include <iostream>
4.  using namespace std;
5.  void main()
6.  {
7.      int num, deci, remainder; //deci is just to store decimal no. temporarily
8.      stack s(100); //assumed stack of 100 int's
9.      cout<<"Enter Bnary Number:"<<endl;
10.     cin>>num;
11.     deci = num; //to display decimal number at the end
12.     do
13.     {
14.         remainder = num%2; //remainder calculation
15.         num = num/2;
16.         s.push(remainder); //pushing to stack
17.     }
18.     while(num>0);
19.
20.     cout<<endl<<endl<<endl<<endl<<endl;
21.     cout<<"================================================================================"<<endl;
22.     cout<<"                                OUTPUT-RESULT                                   "<<endl;
23.     cout<<"================================================================================"<<endl;
24.     cout<<endl;
25.     cout<<"Decimal Number: "<<deci;
26.     cout<<endl<<endl;
27.     cout<<"Binary Number: ";
28.     s.display(); //displaying data from stack i.e. binary number
29.
30.     getch();
31. }
```

Output:



Exercise 2:

Write a program that reads a string (an array of characters) from a text file. Reverse the string using the Stack and write the reversed string to another text file.

Solution:

stack.h File:

```
#pragma once
class stack
{
private:
    char *arrStack;
    int top;
    int size;
public:
    stack(void);
    stack(int);
    bool isEmpty();
    bool isFull();
    void push(char);
    char pop();
    char _top();
    void display();
};
```

stack.cpp File:

```
#include "stack.h"
#include <iostream>
using namespace std;

stack::stack(void)
{
    size = 100;
    arrStack = new char [size];
    top = -1;
}

stack::stack(int size)
{
    this->size = size;
    arrStack = new char [this->size];
    top = -1;
}

bool stack::isEmpty()
{
    if(top == -1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool stack::isFull()
{
```

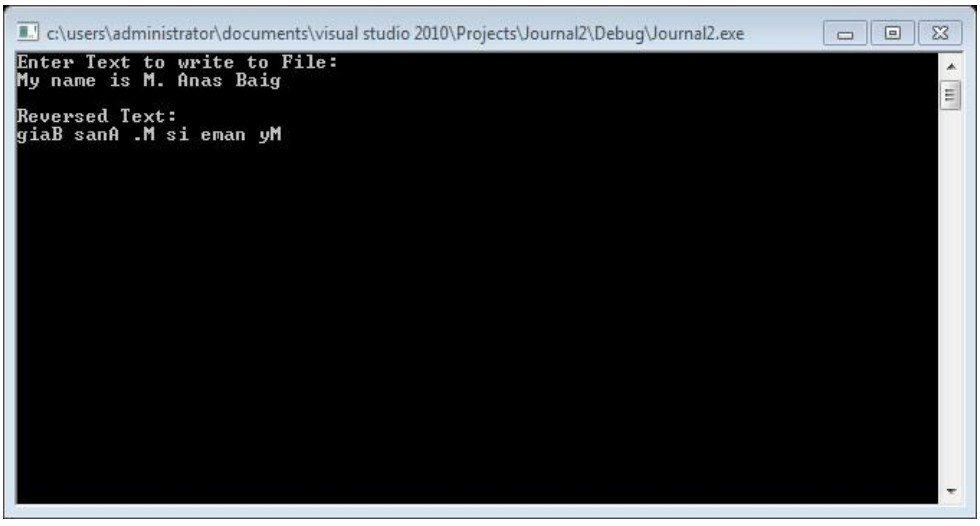
```
33.     if(top == (size-1))
34.     {
35.         return true;
36.     }
37.     else
38.     {
39.     return false;
40.     }
41. }
42.
43. void stack::push(char value)
44. {
45.     if(!isFull())
46.     {
47.         arrStack[++top] = value;
48.     }
49.     else
50.     {
51.         cout<<"Stack Overflow!!!"<<endl;
52.     }
53. }
54.
55. char stack::pop()
56. {
57.     if(!isEmpty())
58.     {
59.         return (arrStack[top--]);
60.     }
61.     else
62.     {
63.         cout<<"Stack Underflow!!!"<<endl;
64.     }
65. }
66.
67. char stack::_top()
68. {
69.     if(!isEmpty())
70.     {
71.         return arrStack[top];
72.     }
73.     else
74.     {
75.         cout<<"Stack Empty!!!";
76.     }
77. }
78.
79. void stack::display()
80. {
81.     for(int i=top; i>=0; i--)
82.     {
83.         cout<<arrStack[i];
84.     }
85. }
```

main.cpp File:

```
1.  #include "stack.h"
2.  #include <fstream>
3.  #include "string"
4.  #include "string.h"
5.  #include "conio.h"
6.  #include <iostream>
7.  using namespace std;
8.
```

```
9.  void main()
10. {
11.     //=========Module to get Text from user and then saving to file===========
12.     string line_write; //string to get input and then write text to file
13.     cout<<"Enter Text to write to File: "<<endl;
14.     getline(cin, line_write); //getting input from user
15.     ofstream write_file ("anas_simple_text.txt"); //write file object
16.     write_file<<line_write; //writing string whole text to file
17.     write_file.close(); //closing write file
18.
19.     //=========Module to read Text from file==================================
20.     string line_read; //string to read text from file
21.     ifstream file_read ("anas_simple_text.txt"); //read file object
22.     getline(file_read,line_read); //reading whole text from file to string
23.     file_read.close(); //closing read file
24.
25.     //=========Module to push all Text to stack to reverse it=================
26.     stack s( line_read.length() ); //declaring stack of the length of the string
27.     for(int i=0; i<( line_read.length() ); i++) //loop to push string characters tostack one by one
28.     {
29.     s.push(line_read[i]); //pushing character from string to stack
30.     }
31.
32.     //=========Module to save reversed text from stack to file================
33.     ofstream write_stack ("anas_reversed_text.txt"); //writing reversed text fromstack to file
34.     do
35.     {
36.     write_stack<<s.pop();
37.     }
38.     while(!s.isEmpty());
39.     write_stack.close();
40.
41.     //=========Module to read Reversed Text from file=========================
42.     string reversed; //string to read reversed text from file
43.     ifstream read_rev ("anas_reversed_text.txt"); //reverse read file object
44.     getline(read_rev,reversed); //reading whole reversed text from file to string
45.     file_read.close(); //closing reversed read file
46.     cout<<endl<<"Reversed Text: "<<endl<<reversed;
47.
48.     getch();
49. }
```

Output:



Exercise 3:

Write a function that reads a Mathematical expression from a text file and verifies the validity of paranthesis in the expression using a Stack.

Solution:

stack.h File:

```
#pragma once
class stack
{
private:
    char *arrStack;
    int top;
    int size;
public:
    stack(void);
    stack(int);
    bool isEmpty();
    bool isFull();
    void push(char);
    char pop();
    char _top();
    void display();
};
```

stack.cpp File:

```
#include "stack.h"
#include <iostream>
using namespace std;

stack::stack(void)
{
    size = 100;
    arrStack = new char [size];
    top = -1;
}

stack::stack(int size)
{
    this->size = size;
    arrStack = new char [this->size];
    top = -1;
}

bool stack::isEmpty()
{
    if(top == -1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool stack::isFull()
{
```

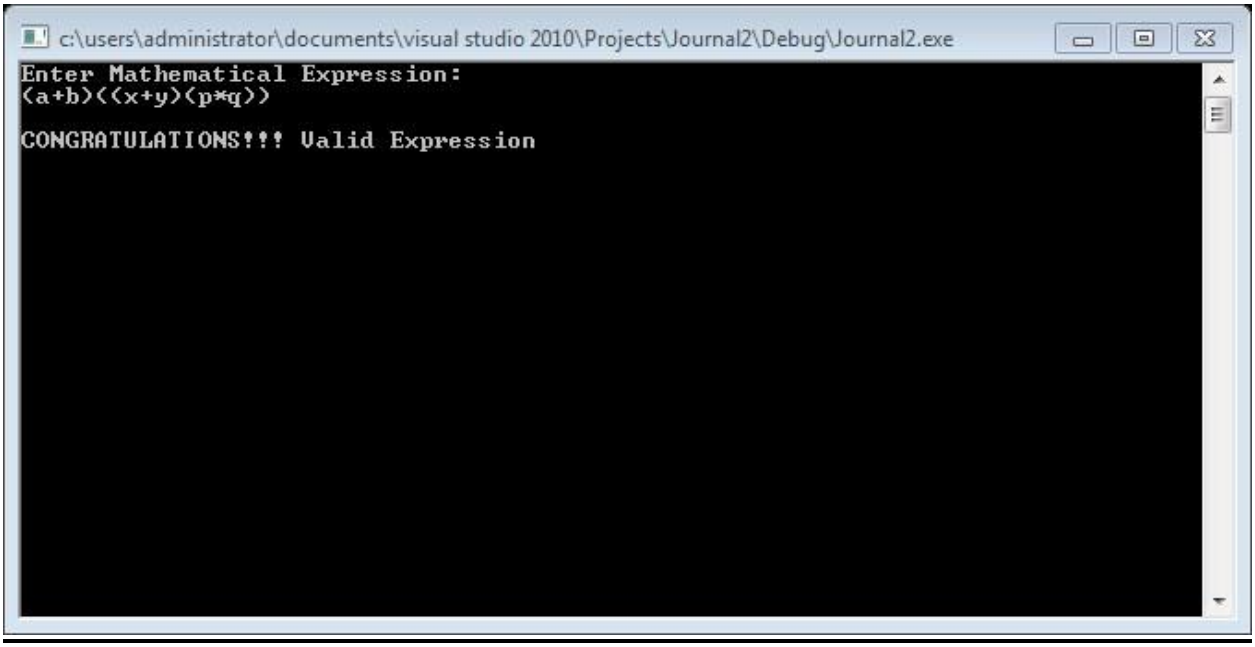
```
33.     if(top == (size-1))
34.     {
35.         return true;
36.     }
37.     else
38.     {
39.         return false;
40.     }
41. }
42.
43. void stack::push(char value)
44. {
45.     if(!isFull())
46.     {
47.         arrStack[++top] = value;
48.     }
49.     else
50.     {
51.         cout<<"Stack Overflow!!!"<<endl;
52.     }
53. }
54.
55. char stack::pop()
56. {
57.     if(!isEmpty())
58.     {
59.         return (arrStack[top--]);
60.     }
61.     else
62.     {
63.         cout<<"Stack Underflow!!!"<<endl;
64.     }
65. }
66.
67. char stack::_top()
68. {
69.     if(!isEmpty())
70.     {
71.         return arrStack[top];
72.     }
73.     else
74.     {
75.         cout<<"Stack Empty!!!";
76.     }
77. }
78.
79. void stack::display()
80. {
81.     for(int i=top; i>=0; i--)
82.     {
83.         cout<<arrStack[i];
84.     }
85. }
```

main.cpp File:

```
1.  #include "stack.h"
2.  #include <fstream>
3.  #include "string"
4.  #include "string.h"
5.  #include "conio.h"
6.  #include <iostream>
7.  using namespace std;
8.
```

```
9.  void main()
10. {
11.     //=========Module to get Mathematical Expression from user and then saving tofile==========
12.     string line_write; //string to get input and then write text to file
13.     cout<<"Enter Mathematical Expression: "<<endl;
14.     getline(cin, line_write); //getting input from user
15.     ofstream write_file ("anas_mathematical_expression.txt"); //write file object
16.     write_file<<line_write; //writing string whole text to file
17.     write_file.close(); //closing write file
18.
19.     //=========Module to read Mathematical Expression fromfile=================================
20.     string line_read; //string to read text from file
21.     ifstream file_read ("anas_mathematical_expression.txt"); //read file object
22.     getline(file_read,line_read); //reading whole text from file to string
23.     file_read.close(); //closing read file
24.
25.     //=========Module to track brackets using stack============================================
26.     stack s(line_read.length());
27.     for(int i=0; i<line_read.length(); i++)
28.     {
29.         if(line_read[i]=='(')
30.         {
31.             s.push('(');
32.         }
33.         if(line_read[i]==')')
34.         {
35.             if(  !(s.isEmpty())  ) //stack must have some elements to pop it
36.             {
37.                 s.pop();
38.             }
39.             else //for invalid expression if there comes closing bracket before opening bracket then it pushes 'e
    ' to make invalid
40.             {
41.                 s.push('e'); //expression to make stack abnormal
42.                 s.push('e'); //expression to make stack abnormal
43.                 s.push('e'); //expression to make stack abnormal
44.             }
45.         }
46.     }
47.     cout<<endl;
48.     if(s.isEmpty())
49.     {
50.         cout<<"CONGRATULATIONS!!! Valid Expression"<<endl;
51.     }
52.     else
53.     {
54.         cout<<"SORRY!!! Invalid Expression"<<endl;
55.     }
56.
57.     getch();
58. }
```

Output:



Exercise 4:

Implement a program to read a postfix expression from a text file, evaluate the expression using a Stack and display the result. The text file should contain expressions in the form as illustrated in the following. (For simplicitiy, assume single digit numbers in the expression.)
23+5*6+

Solution:

stack.h File:

```
1.  #pragma once
2.  class stack
3.  {
4.  private:
5.      int *arrStack;
6.      int top;
7.      int size;
8.  public:
9.      stack(void);
10.     stack(int);
11.     bool isEmpty();
12.     bool isFull();
13.     void push(int);
14.     int pop();
15.     int _top();
16.     void display();
17. };
```

stack.cpp File:

```
1.  #include "stack.h"
```

```
2.  #include <iostream>
3.  using namespace std;
4.
5.  stack::stack(void)
6.  {
7.      size = 100;
8.      arrStack = new int [size];
9.      top = -1;
10. }
11.
12. stack::stack(int size)
13. {
14.     this->size = size;
15.     arrStack = new int [this->size];
16.     top = -1;
17. }
18.
19. bool stack::isEmpty()
20. {
21.     if(top == -1)
22.     {
23.         return true;
24.     }
25.     else
26.     {
27.         return false;
28.     }
29. }
30.
31. bool stack::isFull()
32. {
33.     if(top == (size-1))
34.     {
35.         return true;
36.     }
37.     else
38.     {
39.         return false;
40.     }
41. }
42.
43. void stack::push(int value)
44. {
45.     if(!isFull())
46.     {
47.         arrStack[++top] = value;
48.     }
49.     else
50.     {
51.         cout<<"Stack Overflow!!!"<<endl;
52.     }
53. }
54.
55. int stack::pop()
56. {
57.     if(!isEmpty())
58.     {
59.         return (arrStack[top--]);
60.     }
61.     else
62.     {
63.         cout<<"Stack Underflow!!!"<<endl;
64.     }
65. }
66.
```

```
67. int stack::_top()
68. {
69.     if(!isEmpty())
70.     {
71.         return arrStack[top];
72.     }
73.     else
74.     {
75.         cout<<"Stack Empty!!!";
76.     }
77. }
78. 
79. void stack::display()
80. {
81.     for(int i=top; i>=0; i--)
82.     {
83.         cout<<arrStack[i];
84.     }
85. }
```

main.cpp File:

```
1.  #include "stack.h"
2.  #include <fstream>
3.  #include "string"
4.  #include "string.h"
5.  #include "conio.h"
6.  #include <iostream>
7.  using namespace std;
8.  
9.  void main()
10. {
11.     //=========Module to get Text from user and then saving to file===========
12.     string line_write; //string to get input and then write text to file
13.     cout<<"Enter Postfix Expression to write to File: "<<endl;
14.     getline(cin, line_write); //getting input from user
15.     ofstream write_file ("anas_simple_text.txt"); //write file object
16.     write_file<<line_write; //writing string whole text to file
17.     write_file.close(); //closing write file
18.
19.     //=========Module to read Text from file==================================
20.     string line_read; //string to read text from file
21.     ifstream file_read ("anas_simple_text.txt"); //read file object
22.     getline(file_read,line_read); //reading whole text from file to string
23.     file_read.close(); //closing read file
24.
25.     //=========Module to evaluate postfix expression=========================
26.     stack s(line_read.length()); //declaring stack of te length of the text in a file
27.     for(int i=0; i<line_read.length(); i++) //loop to track each character of string one by one
28.     {
29.         if( isdigit(line_read[i]) ) //condition if the character is numeric digit
30.         {
31.             s.push( line_read[i] - '0'); //converting 'char' to 'int' and then pushing it to string
32.         }
33.         else //condition if the character is other than numeric digit
34.         {
35.             int result;
36.             int operand1 = s.pop();
37.             int operand2 = s.pop();
38.             switch (line_read[i]) //to track the operators
39.             {
40.             case '+': //addition
41.                 {
42.                 result = operand2 + operand1;
```

```
43.                 break;
44.                 }
45.             case '-': //subtraction
46.                 {
47.                     result = operand2 - operand1;
48.                     break;
49.                 }
50.             case '*': //multiplication
51.                 {
52.                     result = operand2 * operand1;
53.                     break;
54.                 }
55.             case '/': //division
56.                 {
57.                     result = operand2 / operand1;
58.                     break;
59.                 }
60.             case '%': //modulus
61.                 {
62.                     result = operand1 % operand2;
63.                     break;
64.                 }
65.             }
66.             s.push(result); //pusing result in to stack
67.         }
68.     }
69.     cout<<endl<<"Result:"<<s._top();
70.
71.     getch();
72. }
```

Output: