Name: **M**uhammad **S**herjeel **A**khtar

Roll **N**o: 20P-0101

Subject: **D**ata **S**tructures **L**ab

Assignment **N**o:10

Submitted **T**o **R**espected **S**ir: **Khurram Shahzad**

Q:3

```cpp
#include <iostream>
using namespace std;
class queue{
    int *arr;
    int size , length , front , rear;
    public:
    queue(int s){
        arr = new int [s];
        size = s;
        front = 0;
        rear = -1;
        length = 0;
    }

    bool isfull(){
        if(front == 0 && rear == size -1){
            return true;
    }
            return false;
}
    void Enqueued(int val)
    {
        if(isfull()){
            cout<<"Your Given Queue Is Overflowing";
}
        else{
            rear = rear+1%size;
            arr[rear] = val;}
}
```

.

```
25  }
26          else{
27              rear = rear+1%size;
28              arr[rear] = val;}
29  }
30      void Dequeued(){
31          if(isEmpty()){
32              cout<<"Your Given Queue Is Empty";
33  }
34          else{
35              front = front+1%size;}
36  }
37  void queueDisplaying()
38  {
39          if(isEmpty())
40              cout<<"Your Given Queue Is Empty";
41          else{
42              int i;
43          if( front <= rear ){
44              for( i=front ; i<= rear ; i++)
45                  cout<<arr[i]<<" ";}
46          else{
47              i=front;
48              while( i < size){
49                  cout<<arr[i]<<" ";
```

.

```
34          else{
35              front = front+1%size;}
36  }
37  void queueDisplaying()
38  {
39          if(isEmpty())
40              cout<<"Your Given Queue Is Empty";
41          else{
42              int i;
43          if( front <= rear ){
44              for( i=front ; i<= rear ; i++)
45                  cout<<arr[i]<<" ";}
46          else{
47              i=front;
48              while( i < size){
49                  cout<<arr[i]<<" ";
50                  i++;}
51                  i=0;
52              while( i <= rear){
53                  cout<<arr[i]<<" ";
54                  i++;}}}
55  }
56  bool isEmpty()
57  {
58          if(front == -1 && rear == -1)
59          return true;
60          else
61          return false;}
```

.

```
39          if(isEmpty())
40              cout<<"Your Given Queue Is Empty";
41          else{
42              int i;
43          if( front <= rear ){
44              for( i=front ; i<= rear ; i++)
45                  cout<<arr[i]<<" ";}
46          else{
47              i=front;
48              while( i < size){
49                  cout<<arr[i]<<" ";
50                  i++;}
51                  i=0;
52              while( i <= rear){
53                  cout<<arr[i]<<" ";
54                  i++;}}}
55  }
56  bool isEmpty()
57  {
58          if(front == -1 && rear == -1)
59          return true;
60          else
61          return false;}
62  };
63  int main(){
64      queue object1(7);
65      object1.Enqueued(8);
```

.

```
46          else{
47              i=front;
48              while( i < size){
49                  cout<<arr[i]<<" ";
50                  i++;}
51                  i=0;
52              while( i <= rear){
53                  cout<<arr[i]<<" ";
54                  i++;}}
55  }
56  bool isEmpty()
57  {
58          if(front == -1 && rear == -1)
59          return true;
60          else
61          return false;}
62  };
63  int main(){
64      queue object1(7);
65      object1.Enqueued(8);
66      object1.Enqueued(4);
67      object1.Enqueued(12);
68      object1.Enqueued(18);
69      object1.Enqueued(2);
70      object1.Dequeued();
71      object1.Dequeued();
72      object1.queueDisplaying();
73      return 0;}
74
```

.

```
63  int main(){
64      queue object1(7);
65      object1.Enqueued(8);
66      object1.Enqueued(4);
67      object1.Enqueued(12);
68      object1.Enqueued(18);
69      object1.Enqueued(2);
70      object1.Dequeued();
71      object1.Dequeued();
72      object1.queueDisplaying();
73      cout<<endl;
74      if(object1.isEmpty()==true){
75          cout<<"Yes Empty";
76      }
77      else{
78          cout<<"Not Empty";
79      }
80      //object1.isfull();
81      return 0;}
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

```
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ cd "/home/spoofy/Downloads/Data Lab"
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ ./"f"
12 18 2
Not Emptyspoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$
```

.

Q2:1

```
1  #include<iostream>…
4  using namespace std;
5  class Stack {
6      int top;
7      int Sizey;
8      int*arr;
9  public:
10     Stack(int size){
11         top = -1;
12         Sizey = size;
13         arr = new int [Sizey];
14     }
15     bool push(int x);
16     int pop();
17     int peek();
18     bool isEmpty();
19     bool isFull();
20 };
21
22 bool Stack::isEmpty(){
23     return (top < 0);
24 }
25 bool Stack::isFull(){
26     return (top == Sizey - 1);
27 }
28 int Stack::peek(){
29     return arr[top];
30 }
31 bool Stack::push(int x) {
```

.

```
22  bool Stack::isEmpty(){
23      return (top < 0);
24  }
25  bool Stack::isFull(){
26      return (top == Sizey - 1);
27  }
28  int Stack::peek(){
29      return arr[top];
30  }
31  bool Stack::push(int x) {
32      if (top >= (Sizey-1)) {
33          cout << "Stack Is Overflowing";
34          return false;
35      }
36      else {
37          top++;
38          arr[top] = x;
39          return true;
40      }
41  }
42  int Stack::pop()
43  {
44      if (top < 0) {
45          cout << "Stack Is Underflowing";
46      }
47      else {
48          int x = arr[top];
49          top--; return x;
50      }
```

.

```
46      }
47      else {
48          int x = arr[top];
49          top--; return x;
50      }
51  }
52
53  int priority (char alpha)
54  {
55      if(alpha == '+' || alpha =='-')
56          return 1;
57      if(alpha == '*' || alpha =='/')
58          return 2;
59      if(alpha == '^')
60          return 3;
61
62      return 0;
63  }
64
65  string convert(string infix)
66  {
67      int i = 0;
68      string postfix = "";
69
70      Stack s(20);
71      while(infix[i]!='\0')
```

.

```
65  string convert(string infix)
66  {
67      int i = 0;
68      string postfix = "";
69
70      Stack s(20);
71      while(infix[i]!='\0')
72      {
73          if(infix[i]>='a' && infix[i]<='z'|| infix[i]>='A'&& infix[i]<='Z')
74          {
75              postfix += infix[i];
76              i++;
77          }
78          else if(infix[i]=='(')
79          {
80              s.push(infix[i]);
81              i++;
82          }
83          else if(infix[i]==')')
84          {
85              while(s.peek()!='(')
86                  postfix += s.pop();
87
88              s.pop();
89              i++;
90          }
91          else
92          {
93              while (!s.isEmpty() && priority(infix[i]) <= priority(s.peek())){
94                  postfix += s.pop();
```

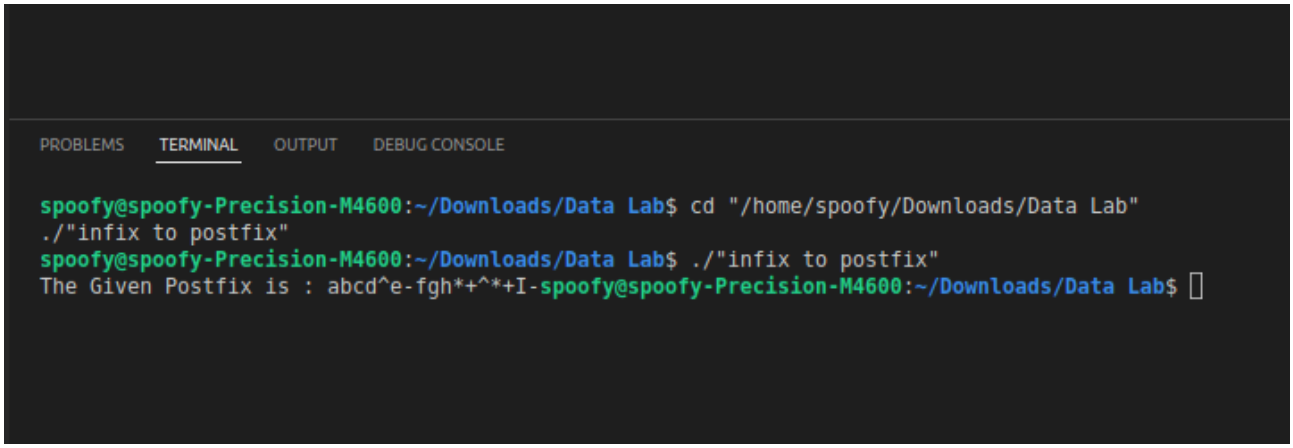
.

```
91          else
92          {
93              while (!s.isEmpty() && priority(infix[i]) <= priority(s.peek())){
94                  postfix += s.pop();
95              }
96              s.push(infix[i]);
97              i++;
98          }
99      }
100     while(!s.isEmpty()){
101         postfix += s.pop();
102     }
103     cout << "Postfix is : " << postfix;
104     return postfix;
105 }
106 int main()
107 {
108     string infix = "a+b*(c^d-e)^(f+g*h)-I";
109     string postfix;
110     cout<<"The Given ";
111     postfix = convert(infix);
112     return 0;
113 }
```

.

```
        else
        {
            while (!s.isEmpty() && priority(infix[i]) <= priority(s.peek())){
                postfix += s.pop();
            }
            s.push(infix[i]);
            i++;
        }
    }
    while(!s.isEmpty()){
        postfix += s.pop();
    }
    cout << "Postfix is : " << postfix;
    return postfix;
}
int main()
{
    string infix = "a+b*(c^d-e)^(f+g*h)-I";
    string postfix;
    cout<<"The Given ";
    postfix = convert(infix);
    return 0;
}
```

.


PROBLEMS
TERMINAL
OUTPUT
DEBUG CONSOLE
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ cd "/home/spoofy/Downloads/Data Lab"
./"infix to postfix"
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ ./"infix to postfix"
The Given Postfix is : abcd^e-fgh*+^*+I-spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$

.

Q2:2

postfix to infix.cpp > ...

```cpp
#include <iostream>
using namespace std;
class StackNode
{
    public: string data;
    StackNode *next;
    StackNode(string data, StackNode *top)
    {
        this->data = data;
        this->next = top;
    }
};
class MyStack
{
    public: StackNode *top;
    int count;
    MyStack()
    {
        this->top = nullptr;
        this->count = 0;
    }
    int size()
    {
        return this->count;
    }
    bool isEmpty()
    {
        if (this->size() > 0)
        {
            return false;
```

.

```
27      bool isEmpty()
28      {
29          if (this->size() > 0)
30          {
31              return false;
32          }
33          else
34          {
35              return true;
36          }
37      }
38      void push(string data)
39      {
40          this->top = new StackNode(data, this->top);
41          this->count++;
42      }
43      string pop()
44      {
45          string temp = "";
46          if (this->isEmpty() == false)
47          {
48              StackNode *t = this->top;
49              temp = this->top->data;
50              this->top = this->top->next;
51              this->count--;
52              delete t;
53          }
54          return temp;
55      }
```

.

```
47          {
48              StackNode *t = this->top;
49              temp = this->top->data;
50              this->top = this->top->next;
51              this->count--;
52              delete t;
53          }
54          return temp;
55      }
56      string peek()
57      {
58          if (!this->isEmpty())
59          {
60              return this->top->data;
61          }
62          else
63          {
64              return "";
65          }
66      }
67  };
68  class Conversion
69  {
70      public:
71          bool isOperator(char text)
72          {
73              if (text == '+' || text == '-' ||
74                  text == '*' || text == '/' ||
75                  text == '^' || text == '%')
76              {
```

.

```
68  class Conversion
69  {
70      public:
71          bool isOperator(char text)
72          {
73              if (text == '+' || text == '-' ||
74                  text == '*' || text == '/' ||
75                  text == '^' || text == '%')
76              {
77                  return true;
78              }
79              return false;
80          }
81      bool isOperands(char text)
82      {
83          if ((text >= '0' && text <= '9') ||
84              (text >= 'a' && text <= 'z') ||
85              (text >= 'A' && text <= 'Z'))
86          {
87              return true;
88          }
89          return false;
90      }
91      void postfixToInfix(string postfix)
92      {
93          int size = postfix.length();
94          MyStack *s = new MyStack();
95          string auxiliary = "";
96          string op1 = "";
97          string op2 = "";
```

.

```
91      void postfixToInfix(string postfix)
92      {
93          int size = postfix.length();
94          MyStack *s = new MyStack();
95          string auxiliary = "";
96          string op1 = "";
97          string op2 = "";
98          bool isValid = true;
99          for (int i = 0; i < size && isValid; i++)
100         {
101             if (this->isOperator(postfix[i]))
102             {
103                 if (s->size() > 1)
104                 {
105                     op1 = s->pop();
106                     op2 = s->pop();
107                     auxiliary = "(" +  op2 + (postfix[i])  +  op1  +  ")";
108                     s->push(auxiliary);
109                 }
110                 else
111                 {
112                     isValid = false;
113                 }
114             }
115             else if (this->isOperands(postfix[i]))
116             {
117                 auxiliary = (postfix[i]);
118                 s->push(auxiliary);
119             }
120             else
```

.

```
                auxiliary = (postfix[i]);
                s->push(auxiliary);
            }
            else
            {
                isValid = false;
            }
        }
        if (isValid == false)
        {
            cout << "Invalid postfix : " << postfix << endl;
        }
        else
        {
            cout << " Postfix : " << postfix << endl;
            cout << " Infix   : " << s->pop() << endl;
        }
    }
};
int main()
{
    Conversion *task = new Conversion();
    string postfix = "ab+c*ef+g/+";
    task->postfixToInfix(postfix);
    postfix = "abc*de-/+";
    task->postfixToInfix(postfix);
    return 0;
}

```

.

```
134         }
135     };
136     int main()
137     {
138         Conversion *task = new Conversion();
139         string postfix = "ab+c*ef+g/+";
140         cout<<"The Infix Notation Is:";
141         task->postfixToInfix(postfix);
142         postfix = "abc*de-/+";
143         cout<<"The Infix Notation Is:";
144         task->postfixToInfix(postfix);
145         return 0;
146     }
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

```
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ cd "/home/spoofy/Downloads/Data Lab"
./"postfix to infix"
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ ./"postfix to infix"
The Infix Notation Is: Postfix : ab+c*ef+g/+
 Infix   : (((a+b)*c)+((e+f)/g))
The Infix Notation Is: Postfix : abc*de-/+
 Infix   : (a+((b*c)/(d-e)))
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$
```

.

Q2:3

prefix to postfix.cpp > ...

```cpp
#include <iostream>
using namespace std;
class StackNode
{
    public: string data;
    StackNode *next;
    StackNode(string data, StackNode *top)
    {
        this->data = data;
        this->next = top;
    }
};
class MyStack
{
    public:
    StackNode *top;
    int count;
    MyStack()
    {
        this->top = nullptr;
        this->count = 0;
    }
    int size()
    {
        return this->count;
    }
    bool isEmpty()
    {
        if (this->size() > 0)
        {
```

.

```
}
        else{
            rear = rear+1%size;
            arr[rear] = val;}
}
    void Dequeued(){
        if(isEmpty()){
            cout<<"Your Given Queue Is Empty";
}
        else{
            front = front+1%size;}
}
void queueDisplaying()
{
        if(isEmpty())
            cout<<"Your Given Queue Is Empty";
        else{
            int i;
        if( front <= rear ){
            for( i=front ; i<= rear ; i++)
                cout<<arr[i]<<" ";}
        else{
            i=front;
            while( i < size){
                cout<<arr[i]<<" ";
```

.

```
82      bool isOperands(char text)
83      {
84          if ((text >= '0' && text <= '9') ||
85              (text >= 'a' && text <= 'z') ||
86              (text >= 'A' && text <= 'Z'))
87          {
88              return true;
89          }
90          return false;
91      }
92      void prefixToPostfix(string prefix)
93      {
94          int size = prefix.length();
95          MyStack *s = new MyStack();
96          string auxiliary = "";
97          string op1 = "";
98          string op2 = "";
99          bool isValid = true;
100         for (int i = size - 1; i >= 0; i--)
101         {
102             if (this->isOperator(prefix[i]))
103             {
104                 if (s->size() > 1)
105                 {
106                     op1 = s->pop();
107                     op2 = s->pop();
```

.

```
104                 if (s->size() > 1)
105                 {
106                     op1 = s->pop();
107                     op2 = s->pop();
108                     auxiliary = op1  +  op2  +  (prefix[i]);
109                     s->push(auxiliary);
110                 }
111                 else
112                 {
113                     isValid = false;
114                 }
115             }
116             else if (this->isOperands(prefix[i]))
117             {
118                 auxiliary = prefix[i];
119                 s->push(auxiliary);
120             }
121             else
122             {
123                 isValid = false;
124             }
125         }
126         if (isValid == false)
127         {
128             cout << "Invalid Prefix : " << prefix << endl;
129         }
130         else
131         {
```

.

```
        {
            cout << "Invalid Prefix : " << prefix << endl;
        }
        else
        {
            cout << " Prefix : " << prefix << endl;
            cout << " Postfix  : " << s->pop() << endl;
        }
    }
};
int main()
{
    Conversion *task = new Conversion();
    string prefix = "*+AB-CD";
    cout<<"After Coversion Notation Is:";
    task->prefixToPostfix(prefix);
    return 0;
}
```

.

Q1:

```cpp
#include<iostream>
using namespace std;
class Queue{
    private:
    int*arr;
    int front,rear;
    int size;
    public:
    Queue(int sized){
        size=sized;
        arr=new int[size];
        front=rear=-1;
    }
    void Enqueue(int valve){
        if(rear<size){
            rear++;
            arr[rear]=valve;
            return;
        }
        if(rear==size){
            rear=front;
            arr[rear]=valve;
            return;
        }
    }
    int Dequeue(){
        int typed;
        typed=arr[front];
        front++;
        return typed;
    }
```

.

```
26      int Dequeue(){
27          int typed;
28          typed=arr[front];
29          front++;
30          return typed;
31      }
32      int Top(){
33          return arr[rear];
34      }
35      bool isFull(){
36          if(rear==size){
37              return true;
38          }
39          return false;
40      }
41      bool isEmpty(){
42          if(rear==front){
43              return true;
44          }
45          return false;
46      }
47      bool onlineCheck(){
48          char check;
49          cout<<"Enter the button for stroke confirmation: \n";
50          cout<<"Press Y or y to activate: \n";
51          cin>>check;
52          if(check=='y'||check=='Y'){
53              return true;
54          }
55          return false;
56      }
```

.

```
47      bool onlineCheck(){
48          char check;
49          cout<<"Enter the button for stroke confirmation: \n";
50          cout<<"Press Y or y to activate: \n";
51          cin>>check;
52          if(check=='y'||check=='Y'){
53              return true;
54          }
55          return false;
56      }
57      void messageBuffer(int valve){
58          for(int i=0;i<size;i++){
59          Enqueue(valve);}
60      }
61      void Buffed(){
62          if(onlineCheck()==true){
63              int x1=Dequeue();
64              int x2=Dequeue();
65              int x3=Dequeue();
66              cout<<"First Message Recieved Is: "<<x1<<endl;
67              cout<<"Second Message Recieved Is: "<<x2<<endl;
68              cout<<"Third Message Recieved Is: "<<x3<<endl;
69              return;
70          }
71          cout<<"The User Is Offline At The Moment";
72      }
73  };
```

.

```
            cout<<"Second Message Recieved Is: "<<x2<<endl;
            cout<<"Third Message Recieved Is: "<<x3<<endl;
            return;
        }
        cout<<"The User Is Offline At The Moment";
    }
};
int main(){
    Queue s1(20);
    s1.Enqueue(5);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Dequeue();
    //int x1=s1.Dequeue();
    //int x2=s1.Dequeue();
    //int x3=s1.Dequeue();
    //int x4=s1.Dequeue();
    //int x5=s1.Dequeue();
    //cout<<x1<<'\n';
    //cout<<x2<<'\n';
    //cout<<x3<<'\n';
    //cout<<x4<<'\n';
```

.

```
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Enqueue(6);
    s1.Dequeue();
    //int x1=s1.Dequeue();
    //int x2=s1.Dequeue();
    //int x3=s1.Dequeue();
    //int x4=s1.Dequeue();
    //int x5=s1.Dequeue();
    //cout<<x1<<'\n';
    //cout<<x2<<'\n';
    //cout<<x3<<'\n';
    //cout<<x4<<'\n';
    //cout<<x5<<'\n';
    //cout<<s1.Top();
    if(s1.isFull()){
        cout<<"Queue is FULL ";
    }
    if(s1.isEmpty()){
        cout<<"Queue is Empty ";
    }
    s1.Buffed();
}
```

.

valla.cpp > main()

```cpp
#include <iostream>
#include <string>
#include <queue>
#include <stack>

using namespace std;
int main()
{
  queue<string> q;


  q.push("First Message Recieved Is : Hi");
  q.push("second Message Recieved Is : How are You");
  q.push("Third Message Recieved Is : What's your name ");

  while(!q.empty()) {

    cout << q.front() << "\n";
    q.pop();
  }
  cout << endl;
  return 0;
}
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

```
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ cd "/home/spoofy/Downloads/Data Lab"
spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$ ./"valla"
First Message Recieved Is : Hi
second Message Recieved Is : How are You
Third Message Recieved Is : What's your name

spoofy@spoofy-Precision-M4600:~/Downloads/Data Lab$
```

.