Instructions: Please read carefully

- Please rename this file as only your ID number **(e.g. 18-*****-1.doc or 18-*****-1.pdf).**
- Submit the file before **25-02-2021** in the Portal Lab Performance section labeled **Lab task 5.** **If you cannot complete the full task, do not worry. Just upload what you have completed.**

Name:- Amit Podder

ID:- 20-42273-1

Section:- [F]

1. Write a code to implement an array (static) based queue and its operations (Enqueue and Dequeue). Get options from the user to Enqueue, Dequeue and display the queue. If the Queue is full and you want to enqueue another value show message "Queue overflowed!!!". If empty show "Queue is empty"

Example:
What you want to do?
1. Enqueue element in the queue
2. Dequeue element from the queue
3. Display the queue

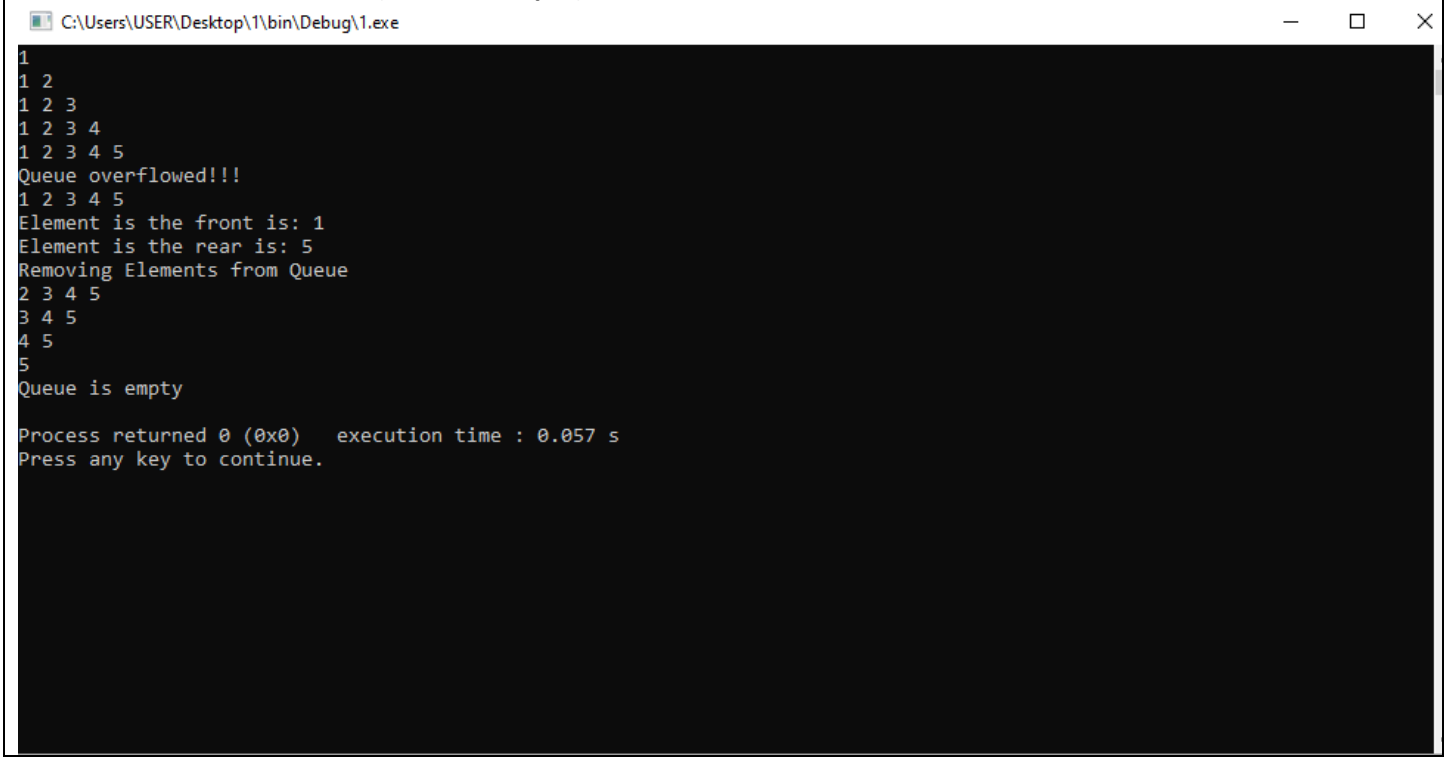
Your code here:

```
#include <iostream>

using namespace std;
#define Size 5
int front=-1;
int rear=-1;

int a[Size];

bool isempty()
{
   if(front==-1&&rear==-1)
      return true;
   else
      return false;
}
void enqueue(int n)
{
   if(rear==Size-1)
      cout<<"Queue overflowed!!!"<<endl;
   else
   {
      if(front==-1)
      front=0;
      rear++;
      a[rear]=n;
   }
}
void dequeue()
```

```
{
   if(isempty())
      cout<<"Queue is empty"<<endl;
   else
   if(front==rear)
      front=rear=-1;
   else
      front++;
}
void sfront()
{
   if(isempty())
      cout<<"Queue is empty"<<endl;
   else
      cout<<"Element is the front is: "<<a[front];
      cout<<endl;
}
void srear()
{
   if(isempty())
      cout<<"Queue is empty"<<endl;
   else
      cout<<"Element is the rear is: "<<a[rear];
      cout<<endl;
}
void display()
{
   if(isempty())
      cout<<"Queue is empty"<<endl;
   else
   {
      for(int i=front;i<=rear;i++)
      cout<<a[i]<<" ";
      cout<<endl;
   }
}

int main()
{
   enqueue(1);display();

   enqueue(2);display();

   enqueue(3);display();

   enqueue(4);display();

   enqueue(5);display();

   enqueue(6);display();

   sfront();
   srear();
```

```
    cout<<"Removing Elements from Queue"<<endl;

    dequeue();display();

    dequeue();display();

    dequeue();display();

    dequeue();display();

    dequeue();display();

    return 0;
}
```

Your whole Screenshot here: (Console Output):



Code Instruction:

For both of the following problems, an operand is assumed to be a single digit. And an operator is limited to '**+**', '**-**', '*****', '**/**' (these 4 types). Also, for usage of parentheses, use only '(' for opening and ')' for closing.

In light of these remarks, an algebraic expression for example can be written like below:

2 * 4 + (6 – 3) / 3

2. Write a code to convert an infix algebraic expression to both postfix and prefix using the help of Stack or Queue.

Your code here:

```
#include <iostream>
#include<stack>
```

```
using namespace std;
int p(char c)
{
  if(c=='^')
  return 3;
  else if(c=='*'||c=='/')
  return 2;
  else if(c=='+'||c=='-')
  return 1;
  else
   return -1;
}

void pf(string sr)
{
      std::stack<char>su;
      su.push('N');

      int a=sr.length();
      string sr2;

      for(int k=0;k<a;k++)
      {
         if(sr[k]=='0'&&sr[k]=='9')
            sr2+=sr[k];
         else if(sr[k]=='(')
                  su.push('(');
         else if(sr[k]==')')
         {
            while(su.top()!='N'&& su.top()!='(')
            {
               char c=su.top();
               su.pop();
               sr2+=c;
            }
            if(su.top()=='(')
            {
               char c =su.top();
               su.pop();
            }
         }
      else
         {
               while(su.top()!='N'&& p(sr[k])<=p(su.top()))
               {
               char c =su.top();
               su.pop();
               sr2+=c;
               }
               su.push(sr[k]);
         }
      }
      while(su.top()!='N')
```

```
        {
            char c =su.top();
            su.pop();
            sr2+=c;
        }

    cout<<"The postfix expression: "<<sr2<<endl;
}

int main()
{
  string l="2*4+(6-3)/3";
  pf(l);
  return 0;
}
```

Your whole Screenshot here: (Console Output):