DSA- Worksheet 5

Name: Sucheta Pal UID: 20BCS7901

Q1>Implement Queue using Stacks

Code

```
class MyQueue {

public:

stack<int> input;

 stack<int> output;

 MyQueue() {

 }

 void push(int x) {

 input.push(x);

 }

 int pop() {

 int x;

 if(output.empty()==false){

 x = output.top();
 output.pop();
 return x; }
 else{
  while(input.empty()==false)
{  output.push(input.top());
  input.pop();
 }
```

```
 x = output.top();
 output.pop();
 return x;
 }
 }
 int peek() {
 if(output.empty()==false)
return output.top();
 else{
 while(input.empty()==false){
output.push(input.top());
input.pop();
}
                                bool empty() {
return output.top(); }  }
```

```
return output.empty() && input.empty() ;  }
 };
```

Q2>Min Stack Code

```
Class MinStack { public: vector< pair<int,int> > s;
MinStack() { }
void push(int val) {
if(s.empty())
s.push_back({val,val});
else
s.push_back({val,min(s.back().second,val)}); }

void pop()
{ s.pop_back(); }
int top()
{ return s.back().first; }
int getMin()
{ return s.back().second; }
};
```

Q3>First Unique Character in a String

Code

```
class Solution {
public:
                    int firstUniqChar(string s) {
                    int arr[256]={0};
```

```
                                return i;
                                }
for(int i=0;i<s.size();i++)     }
{                               return -1;
arr[s[i]]++;                    }
}                               };
for(int i=0;i<s.size();i++)
{
if(arr[s[i]]==1)
{
```

Q4>Product of the Last k Numbers

Code

```
class ProductOfNumbers {
public:
 vector<int>v;
 ProductOfNumbers() {
 }

                        void add(int num) {
if(num==0) {
v.clear();
 return;
 }
```

```
if(v.empty()) {









v.push_back(num);
}
else {
v.push_back(v.back()*num); }
}
int getProduct(int k) {
if(k>v.size()) {
return 0;
}
if(k==v.size()) {
return v[k-1];
}
return v.back()/(v[v.size()-k-1]);




 }
};
```

Q6>Design Front Middle Back Queue

Code

```
class FrontMiddleBackQueue {
public:
 vector<int> vec;
 FrontMiddleBackQueue() {
                                                    }


 void pushFront(int val) {
vec.insert(vec.begin(),val);
 }
 void pushMiddle(int val) {
 vec.insert(vec.begin()+vec.size()/2,val); }
 void pushBack(int val) {
 vec.push_back(val);
 }
 int popFront() {
 if(vec.size() == 0)
 return -1;
 int value = vec[0];
 vec.erase(vec.begin());
 return value;
 }
 int popMiddle() {
```

```
                    if(vec.size() == 0)
                    return -1;



 int value = vec[(vec.size()-1)/2];
  vec.erase(vec.begin()+(vec.size()-1)/2);
return value;
 }
 int popBack() {
 if(vec.size() == 0)
 return -1;
 int value = vec[vec.size() -1];
 vec.pop_back();
                        return value;
                         }
                        };
```

Q8>Design Circular Queue

Code

```
class MyCircularQueue {
```

```
public:
 int *arr;
 int front;
 int rear;




int size;
 MyCircularQueue(int k) {
 arr = new int[k];
 front = -1;
rear = -1;
 size = k;
 }
 bool enQueue(int value) {
 if(isFull()) return false;
 if(isEmpty()) front = 0;
 rear = (rear + 1) % size;
arr[rear] = value;
 return true;
 }
 bool deQueue() {
 if(isEmpty()) return false; if(front ==
rear) front = rear = -1; else front =
(front + 1) % size; return true;
```

```
}

int Front() {                                   if(isEmpty())
 return -1;
 return arr[front];

 }

 int Rear() {
 if(isEmpty()) return -1;

 return arr[rear];

 }

 bool isEmpty() {

 return front == -1;

 }

 bool isFull() {

                                  return ((rear

                                 + 1) % size)

                                 == front; }

}
```

Q11>Validate Stack Sequences

Code

```
class Solution {

public:

                                      bool validateStackSequences(vector<int>& pushed,
```

```
                                    vector<int>& popped) { int pushId = -1, poppId = 0;



 for (int i = 0; i < pushed.size(); ++i) {
 pushed[++pushId] = pushed[i];
 while (pushId >= 0 and pushed[pushId] == popped[poppId]) { --
pushId;
 ++poppId;
 }
 }
 return pushId == -1;
 }
 };
```