QUEUE AND DEQUE

Q1)implementation of queue

using array:

we use the circular array for representation as deque:

```
//intially cap mentioned
//size=0 front=0
bool isfull(){
   return (size==cap);
}
bool isempty(){
   return (size==0);
}
int getfront(){
   if(isempty){
      return -1;
   }
   return front;
}
int getrear(){
   if(isempty){
      return -1;
   }
  return (front+size-1)%cap;
}
void enque(int x){
   if(isfull){
      return;
   }
   int rear=getrear();
   rear=(rear+1)%cap;
   arr[rear]=x;
   size++;
}
void deque(int x){
   if(isfull){
      return;
   }
   int front=getfront();
   front=(front+1)%cap;
   size--;
}
```

STL :

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int main(){
  ll n,m,a,b;
  queue<ll>q;
  cin>>n;
  for(ll i=0;i<n;i++){
     cin>>m;
     q.push(m);
  }
  cout<<q.size()<<endl;
  cout << q.front() << " " << q.back()<< endl;
   q.pop();
   cout << q.front() << " " << q.back() << endl;
  while(q.empty() == false){
     cout << q.front() << " " << q.back() << endl;
     q.pop();
  }
}
```

Q2)Implementing stack using queue

```
struct stack{
   queue<ll>q,q2;
   int top(){
      return q.front();
   }
   int size(){
      return q.size();
   }
   int pop(){
      if(q.empty()==true){
         return;
      }
      int k=q.front();
      q.pop();
      return k;
   }
   int push(int x){
      while(q.empty()==false){
         q2.push(q.front());
         q.pop();
      }
      q.push(x);
       while(q2.empty()==false){
         q.push(q2.front());
         q2.pop();
      }
   }
}
```

Q3)reverse a queue

Put the thing in the stack and reverse it..

```
#include <bits/stdc++.h>
#include <queue>
using namespace std;


void Print(queue<int>& Queue)
{
            while (!Queue.empty()) {
```

```
                cout << Queue.front() << " ";
                Queue.pop();
        }
}
void reverseQueue(queue<int>& Queue) {
        stack<int> Stack;
        while (!Queue.empty()) {
                Stack.push(Queue.front());
                Queue.pop();
        }
        while (!Stack.empty()) {
                Queue.push(Stack.top());
                Stack.pop();
        }
}
int main() {
        queue<int> q;
        q.push(12);
        q.push(5);
        q.push(15);
        q.push(20);

        reverseQueue(q);
        Print(q);
}
```

Q4)generate numbers with given digits in increasing order

```
#include <bits/stdc++.h>
#include <queue>
using namespace std;


void printFirstN(int n) {
   queue<string> q;

   q.push("5");
   q.push("6");

   for(int i = 0; i < n; i++){
      string curr = q.front();
      cout << curr << " ";
      q.pop();
      q.push(curr + "5");
      q.push(curr + "6");
   }

}


int main()
{
        int n;
        cin>>n;
        printFirstN(n);
}
```

DEQUE:

Q5)Implementation of deque

Using array:

Circular implementation:

```
void deletfront(){
   if(isempty()){
      return;
   }
   front=(front+1)%cap;
   size--;
}
void insertrear(){
   if(inempty()){
      return;
   }
   int new_rear=(front+size)%cap;
   arr[new_rear]=n;
   size++;
}
void insertfront(int n){
   if(isfull()){
      return;
   }
   front=(front+cap-1)%cap;
   arr[front]=x;
   size++;
}
void deleterear(){
   if(isempty()){
      return;
   }
   size--;
}
```

Stl approach:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int main(){
   deque<ll>dq;
   ll n,m,p;
   cin>>n;
   for(ll i=0;i<n;i++){
      cin>>m;
      dq.push_back(m);
   }
```

```
   dq.push_front(10);
   for(auto x:dq){
     cout<<x<<" ";
   }
   //insert in the deque:
   auto it=dq.begin();
   it++;
   dq.insert(it,40);
   for(auto x:dq){
      cout<<x;
   }
   cout<<dq.size();
   dq.pop_front();
   dq.pop_back();
   cout<<dq.size();
}
```

Q6)Maximum of all sub-arrays of size k

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
void printMax(vector<ll>arr, int n, int k){
   deque<int> dq;
   for (int i=0;i<k;i++) {
      while (!dq.empty() && arr[i] >=
arr[dq.back()])
         dq.pop_back();
         dq.push_back(i);
   }
   for (int i=k; i < n; ++i) {
      cout << arr[dq.front()] << " ";
      while ((!dq.empty()) && dq.front() <= i -
k)
         dq.pop_front();
      while ((!dq.empty()) && arr[i] >=
arr[dq.back()])
         dq.pop_back();
      dq.push_back(i);
   }
  cout << arr[dq.front()];
}
int main(){
   ll a,b,m,n,k;
   cin>>n>>k;
   vector<ll>v;
   for(ll i=0;i<n;i++){
      cin>>m;
      v.push_back(m);
   }
   printMax(v,n,k);
}
```

Q7)First circular tour

```
int firstpetrol(int petrol[],int dist[],int n){
   int start=0,curr_pet=0,prev_pet=0;
   for(ll i=0;i<n;i++){
      curr_pet+=(petrol[i]-dist[i]);
      if(curr_pet<0){
         start=i+1;
         prev_pet+=curr_pet;
         curr_pet=0;
      }
   }
   return ((curr_pet+prev_pet>=0))?(start+1);
}
```