Saransh Kalra

R11528524

Lab 04

**Header of stack**

#ifndef Stack\_hpp

#define Stack\_hpp

#include <stdio.h>

*//creating the node class for the node of the stack*

**class** Node{

**private**:

*//variable info for storing node info*

**int** info;

*//pointer of class node, pointing to the next node of the stack*

Node \*next;

**public**:

*//constructor and deconstructor for node*

Node(**int**);

~Node();

*//getters and setters for info*

**int** get\_info();

**void** set\_info(**int**);

*//getters and setters for next*

Node\* get\_next();

**void** set\_next(Node\*);

};

**class** Stack{

*//pointer to point of the head of the stack*

Node \*head;

**public**:

*//constructor and deconstructor for the stack*

Stack();

~Stack();

*//function to push stuff on head of the stack*

**void** stack\_push(**int**);

*//function to pop stuff from the head of the stack*

**int** stack\_pop();

*//function to peek onto the top of the stack*

**int** stack\_peek();

*//function to print the stack*

**void** stack\_print();

*//function to check if the stack is empty*

**void** stack\_isEmpty();

};

#endif */\* Stack\_hpp \*/*

***Cpp of stack***

#include "Stack.hpp"

#include <iostream>

**using** **namespace** std;

*//constructor for node class*

Node::Node(**int** info){

*//set info to the info of the node*

**this**->info = info;

*//next of the node is set to null*

**this**->next = 0;

}

*//no need of deconstructor as stack deconstructor handles these*

Node::~Node(){

}

*//getter for the info of the class node*

**int** Node::get\_info(){

**return** **this**->info;

}

*//setter for the info of the class node*

**void** Node::set\_info(**int** info){

**this**->info = info;

}

*//getter for the next of the node class*

Node\* Node::get\_next(){

**return** **this**->next;

}

*//setter for the next of the node class*

**void** Node::set\_next(Node \*next){

**this**->next = next;

}

*//constructor for the stack class*

Stack::Stack(){

**this**->head=0;

}

*//deconstructor for the stack class*

Stack::~Stack(){

Node \*temp = **this**->head;

**while**(temp!=**NULL**){

temp = temp->get\_next();

**delete** temp;

}

}

*//push function for stack class*

**void** Stack::stack\_push(**int** info){

*//new node to be added with info as the argument*

Node \*newNode = **new** Node(info);

*//we are pushing it on the head, so make the next of new node to be head*

newNode->set\_next(**this**->head);

*//make the new node as the new head*

**this**->head = newNode;

}

*//pop function for stack class*

**int** Stack::stack\_pop(){

*//error if the list is empty*

**if** (!**this**->head) {

cout<<"Stack is empty! no of items avilable to pop: "<<endl;

**return** 0;

}

**else**{

*//take a temporary node intialise to head*

Node \*temp = **this**->head;

*//the head is now the next of the head*

**this**->head = temp->get\_next();

*//take a variable to store the popped value*

**int** info = temp->get\_info();

*//delete the deleted node*

**delete** temp;

cout<<"Item popped: ";

*//return the value*

**return** info;

}

}

*//peek function for stack*

**int** Stack::stack\_peek(){

*//if head is not null, return the head as the top value on the stack*

**if**(**this**->head){

cout<<"Last Item in stack: ";

**return** **this**->head->get\_info();

}

*//else error message that stack is empty*

**else** {

cout<<"Stack is empty! Items in Stack: ";

**return** 0;

}

}

*//function for printing the stack*

**void** Stack::stack\_print(){

*//if head is NULL, error stack is empty*

**if**(!**this**->head){

cout<<"stack is empty!"<<endl;

}

**else**{

*//create a temp node for traversing starting for head*

Node \*temp = **this**->head;

cout<<"{ ";

*//while temp is not null*

**while**(temp){

cout<<temp->get\_info()<<"->";

temp = temp->get\_next();

}

cout<<" }"<<endl;

}

}

*//function to check if the stack is empty*

**void** Stack::stack\_isEmpty(){

**if**(**this**->head){

*//if head is not null its not empty*

cout<<"no! stack isn't empty "<<endl;

}

**else**{

cout<<"yes! the stack is empty "<<endl;

}

}

**Main for stack**

#include <iostream>

#include "Stack.hpp"

**using** **namespace** std;

**int** main() {

*//stack for testing*

Stack s1;

*//initialising a choice variable*

**char** choice = '1';

*//while the choice is not the exit choice display the menu again and again*

**while**(choice!='6'){

cout<<"STACK MENU"<<endl;

cout<<"----------"<<endl;

cout<<"1. Push a number on top of stack"<<endl;

cout<<"2. Pop a number from the top of the stack"<<endl;

cout<<"3. Check if the stack is empty"<<endl;

cout<<"4. Take a peek in the stack"<<endl;

cout<<"5. Print the stack"<<endl;

cout<<"6. Exit"<<endl;

cout<<"Enter your choice: "<<endl;

cin>>choice;

*//switch case for menu*

**switch** (choice) {

*//menu option 1 for pushing into stack*

**case** '1':

**int** info;

*//user input for the number they want to put in*

cout<<"enter the number you want to push: "<<endl;

cin>>info;

s1.stack\_push(info);

**break**;

*//menu option 2 for popping from the stack*

**case** '2':

cout<<s1.stack\_pop()<<endl;

**break**;

*//menu option 3 to check if the stack is empty*

**case** '3':

s1.stack\_isEmpty();

**break**;

*//menu option 4 to peek at the top of the stack*

**case** '4':

cout<<s1.stack\_peek()<<endl;

**break**;

*//menu option 5 to print the stack*

**case** '5':

s1.stack\_print();

**break**;

*//menu option 6 to exit*

**case** '6':

**break**;

*//default being an invalid choice*

**default**:

cout<<"invalid choice"<<endl;

**break**;

}

}

**return** 0;

}

**Header for queue**

#ifndef queue\_hpp

#define queue\_hpp

#include <stdio.h>

*//creating the node class for the node of the stack*

**class** Node{

**private**:

*//variable info for storing node info*

**int** info;

*//pointer of class node, pointing to the next node of the stack*

Node \*next;

**public**:

*//constructor and deconstructor for node*

Node(**int**);

~Node();

*//getters and setters for info*

**int** get\_info();

**void** set\_info(**int**);

*//getters and setters for next*

Node\* get\_next();

**void** set\_next(Node\*);

};

**class** Queue{

*//pointer to point of the head and tail of the queue*

Node \*head, \*tail;

**public**:

*//constructor and deconstructor for the queue*

Queue();

~Queue();

*//function to add on head of the queue*

**void** queue\_enqueue(**int**);

*//function to remove from the tail of the queue*

**int** queue\_dequeue();

*//function to peek onto the head of the queue*

**int** queue\_peek();

*//function to print the queue*

**void** queue\_print();

*//function to check if the queue is empty*

**void** queue\_isEmpty();

};

#endif */\* queue\_hpp \*/*

***Cpp for queue***

#include "queue.hpp"

#include <iostream>

**using** **namespace** std;

*//constructor for node class*

Node::Node(**int** info){

*//set info to the info of the node*

**this**->info = info;*//next of the node is set to null*

**this**->next = 0;

}

*//no need of deconstructor as queue deconstructor handles these*

Node::~Node(){

}

*//getter for the info of the class node*

**int** Node::get\_info(){

**return** **this**->info;

}

*//setter for the info of the class node*

**void** Node::set\_info(**int** info){

**this**->info = info;

}

*//getter for the next of the node class*

Node\* Node::get\_next(){

**return** **this**->next;

}

*//setter for the next of the node class*

**void** Node::set\_next(Node \*next){

**this**->next = next;

}

*//constructor for the queue class*

Queue::Queue(){

**this**->head=0;

**this**->tail=0;

}

*//deconstructor for the queue class*

Queue::~Queue(){

Node \*temp = **this**->head;

**while**(temp!=**NULL**){

temp = temp->get\_next();

**delete** temp;

}

}

*//enqueue function for the queue class*

**void** Queue::queue\_enqueue(**int** info){

*//new node to be added with info as the argument*

Node \*newNode = **new** Node(info);

*//we are pushing it on the head, so make the next of new node to be head*

newNode->set\_next(**this**->head);

*//if head is null and new node is the first node then the new node is also the tail*

**if**(!**this**->head) {

**this**->tail = newNode;

}

*//the new head is the new node*

**this**->head = newNode;

}

**int** Queue::queue\_dequeue(){

*//if tail is null, queue is empty give error*

**if**(!**this**->tail) {

cout<<"Queue is empty! no. of items avilable to dequeue: "<<endl;

**return** 0;

}

**else**{

*//if the next of head is null, there is only one element which will be deleted*

**if**(**this**->head->get\_next()==**NULL**){

*//temp for storing the value that is being dequeued*

**int** temp = **this**->tail->get\_info();

*//deleting the node at tail*

**delete** **this**->tail;

cout<<"The value being popped is: "<<endl;

*//as this was the last element make the head and tail point to null making teh list empty*

**this**->head=**this**->tail=**NULL**;

**return** temp;

}

**else**{

cout<<"The value being popped is: "<<endl;

*//else make a tranversal pointer with starts as head*

Node \*ptr = **this**->head;

*//traverse till it doesn't reach the node previous to tail*

**while**(ptr->get\_next()!=**this**->tail){

ptr=ptr->get\_next();

}

*//temp for storing the value that is being dequeued*

**int** temp = **this**->tail->get\_info();

*//delete the node at tail*

**delete** **this**->tail;

*//setting the traversed pointer as teh new tail*

**this**->tail=ptr;

*//setting its next to be null as its the end of teh queue*

**this**->tail->set\_next(**NULL**);

*//returning the removed value*

**return** temp;

}

}

}

*//peek function for queue*

**int** Queue::queue\_peek(){

*//if tail is not null, return the tail as the last value on the queue*

**if**(**this**->tail){

cout<<"Last Item in queue: ";

**return** **this**->head->get\_info();

}

*//else error message that stack is empty*

**else** {

cout<<"queue is empty! Items in Queue: ";

**return** 0;

}

}

*//function for printing the queue*

**void** Queue::queue\_print(){

*//if head is NULL, error queue is empty*

**if**(!**this**->head){

cout<<"queue is empty!"<<endl;

}

**else**{

*//create a temp node for traversing starting for head*

Node \*temp = **this**->head;

cout<<"{ ";

*//while temp is not null*

**while**(temp){

cout<<temp->get\_info()<<"->";

temp = temp->get\_next();

}

cout<<" }"<<endl;

}

}

*//function to check if the queue is empty*

**void** Queue::queue\_isEmpty(){

*//if head is not null its not empty*

**if**(**this**->head){

cout<<"no! queue isn't empty. "<<endl;

}

**else**{

cout<<"yes! the queue is empty. "<<endl;

}

}

**Main for queue**

#include <iostream>

#include "queue.hpp"

**using** **namespace** std;

**int** main() {

*//queue for testing*

Queue q1;

*//initialising a choice variable*

**char** choice = '1';

*//while the choice is not the exit choice display the menu again and again*

**while**(choice!='6'){

cout<<"QUEUE MENU"<<endl;

cout<<"----------"<<endl;

cout<<"1. Push a number on end of queue"<<endl;

cout<<"2. Pop a number from the start of the queue"<<endl;

cout<<"3. Check if the queue is empty"<<endl;

cout<<"4. Take a peek in the queue"<<endl;

cout<<"5. Print the queue"<<endl;

cout<<"6. Exit"<<endl;

cout<<"Enter your choice: "<<endl;

cin>>choice;

*//switch case for menu*

**switch** (choice) {

*//menu option 1 for enqueue into queue*

**case** '1':

**int** info;

*//user input for the number they want to put in*

cout<<"enter the number you want to push: "<<endl;

cin>>info;

q1.queue\_enqueue(info);

**break**;

*//menu option 2 for dequeue from the queue*

**case** '2':

cout<<q1.queue\_dequeue()<<endl;

**break**;

*//menu option 3 to check if the queue is empty*

**case** '3':

q1.queue\_isEmpty();

**break**;

*//menu option 4 to peek at the tail of the queue*

**case** '4':

cout<<q1.queue\_peek()<<endl;

**break**;

*//menu option 5 to print the queue*

**case** '5':

q1.queue\_print();

**break**;

*//menu option 6 to exit*

**case** '6':

**break**;

*//default being an invalid choice*

**default**:

cout<<"invalid choice"<<endl;

**break**;

}

}

**return** 0;

}

**queue output**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

**Program ended with exit code: 0**2

**Queue is empty! no. of items avilable to dequeue:**

**0**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

3

**yes! the queue is empty.**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

4

**queue is empty! Items in Queue: 0**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

1

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

2

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 2->1-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

3

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 3->2->1-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

2

**The value being popped is:**

**1**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 3->2-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

2

**The value being popped is:**

**2**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 3-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

4

**Last Item in queue: 3**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

3

**no! queue isn't empty.**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

6

**Stack output**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

**Program ended with exit code: 0**2

**Stack is empty! no of items avilable to pop:**

**0**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

3

**yes! the stack is empty**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

4

**Stack is empty! Items in Stack: 0**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

1

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

2

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

5

**{ 2->1-> }**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

3

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

5

**{ 3->2->1-> }**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

2

**Item popped: 3**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

5

**{ 2->1-> }**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

2

**Item popped: 2**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

5

**{ 1-> }**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

3

**no! stack isn't empty**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

4

**Last Item in stack: 1**

**STACK MENU**

**----------**

**1. Push a number on top of stack**

**2. Pop a number from the top of the stack**

**3. Check if the stack is empty**

**4. Take a peek in the stack**

**5. Print the stack**

**6. Exit**

**Enter your choice:**

6

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

**Program ended with exit code: 0**2

**Queue is empty! no. of items avilable to dequeue:**

**0**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

3

**yes! the queue is empty.**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

4

**queue is empty! Items in Queue: 0**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

1

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

2

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 2->1-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

1

**enter the number you want to push:**

3

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 3->2->1-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

2

**The value being popped is:**

**1**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 3->2-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

2

**The value being popped is:**

**2**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

5

**{ 3-> }**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

4

**Last Item in queue: 3**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

3

**no! queue isn't empty.**

**QUEUE MENU**

**----------**

**1. Push a number on end of queue**

**2. Pop a number from the start of the queue**

**3. Check if the queue is empty**

**4. Take a peek in the queue**

**5. Print the queue**

**6. Exit**

**Enter your choice:**

6