ASSIGNMENT-4

Name: Prabhnoor Singh Roll No: 102115059 Group:2NC3

Q1. Implement the following stack operations using arrays:

* Push
* Pop
* IsFull
* isEmpty
* Peek
* stackTop

Create a stack structure and stack class for the above said implementation.

#include <iostream> using namespace std; class stacks{ private:

int arr[10]; int top=-1; public:

void push(int num){ top=top+1; arr[top]=num;

}

int pop(){

int temp; temp=arr[top]; top=top-1; return temp;

}

bool isFull(){ if(top==9) return 1; else return 0;

}

bool isEmpty(){ if(top==-1) return 1; else return 0;

}

void peek(){ if(top>-1)

cout<<"Number is :"<<arr[top]<<endl;

else

cout<<"Invalid"<<endl;

}

int stackTop(){ return arr[top];

}

};

int main(){ int ch,c=1,num,pos; stacks s; while(c==1){ cout<<"1. Push"<<endl; cout<<"2. Pop"<<endl; cout<<"3. Is Full"<<endl; cout<<"4. Is Empty"<<endl; cout<<"5. Peek"<<endl; cout<<"6. Stack Top"<<endl; cout<<"Choice: "; cin>>ch; switch(ch){ case 1:

if(s.isFull()) cout<<"Stack is full"<<endl; else{

cout<<"Enter Number: "; cin>>num;

s.push(num);

} break; case 2:

num=s.pop(); cout<<"Number is: "<<num<<endl; break; case 3:

if(s.isFull()) cout<<"Stack is full"<<endl;

else

cout<<"Stack is not full"<<endl; break; case 4:

if(s.isEmpty()) cout<<"Stack is empty"<<endl;

else

cout<<"Stack is not Empty"<<endl; break; case 5:

s.peek(); break; case 6:

if(s.isEmpty()) cout<<"Stack is empty"<<endl;

else{ num=s.stackTop(); cout<<"Top is: "<<num<<endl;

} break; default:

cout<<"Invalid Choice"<<endl;

}

cout<<"Press 1 to continue: "; cin>>c;

}

return 0;

}



Q2. Implement the following stack operations using linked

List:

* Push
* Pop
* IsFull
* isEmpty
* Peek

Create a stack structure and stack class for the above said implementation.

#include <iostream> using namespace std; struct arr{ int data; struct arr \*next;

};

typedef struct arr node; node \*head=NULL, \*top; class stacks{ int max=-1; public: void push(){ int num; max=max+1; if (head!=NULL){ top->next=new node; cout<<"Enter Number: "; cin>>num; top->next->data=num; top->next->next=NULL; top=top->next;

} else{ head = new node; cout<<"Enter Number: "; cin>> num; head->data = num; head->next=NULL; top=head;

}

}

void pop(){ max=max-1; if(head->next==NULL){ cout<<"Number is: "<<head->data<<endl; delete head;

} else{ node \*p=head; while(p->next->next!=NULL){

p=p->next;

}

top=p;

cout<<"Number is: "<<top->next->data<<endl; delete p->next; p->next=NULL;

}

}

bool isEmpty(){ if(head==NULL) return 1; else return 0;

}

bool isFull(){ if(max>=9) return 1; else return 0;

}

void peek(){

cout<<"Number is: "<<top->data<<endl;

}

};

int main(){ int ch,c=1; stacks s; while(c==1){ cout<<"1. Push"<<endl; cout<<"2. Pop"<<endl; cout<<"3. Is Full"<<endl; cout<<"4. Is Empty"<<endl; cout<<"5. Peek"<<endl; cout<<"Choice: "; cin>>ch; switch(ch){ case 1:

if(s.isFull()) cout<<"Stack is full"<<endl; else

s.push(); break; case 2:

if(s.isEmpty()) cout<<"Stack is empty"<<endl;

else

s.pop(); break; case 3:

if(s.isFull()) cout<<"Stack is full"<<endl;

else

cout<<"Stack is not full"<<endl; break; case 4:

if(s.isEmpty()) cout<<"Stack is empty"<<endl;

else

cout<<"Stack is not Empty"<<endl; break; case 5:

s.peek(); break; default:

cout<<"Invalid Choice"<<endl;

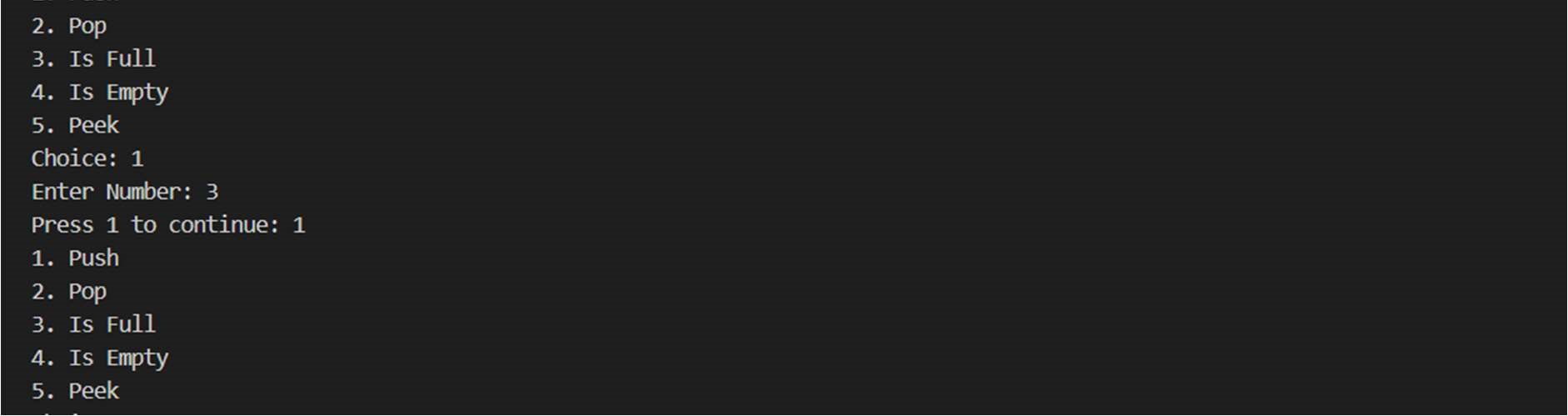
}

cout<<"Press 1 to continue: "; cin>>c;

}

return 0;

}



Q3. Implement linear queue by writing a class and also implement the following functions:

* Insert or enqueue
* Remove or dequeue
* Isfull
* Isempty

Write a main function to exemplify the results. Also write a main function to make the implementation a “Menu-Driven”.

#include <iostream> using namespace std; struct arr{ int data; struct arr \*next;

};

typedef struct arr node; node \*head=NULL, \*top; class queues{ int max=-1; public: void push(){ int num; max=max+1; if (head!=NULL){ top->next=new node; cout<<"Enter Number: "; cin>>num; top->next->data=num; top->next->next=NULL; top=top->next;

} else{ head = new node; cout<<"Enter Number: "; cin>> num; head->data = num; head->next=NULL; top=head;

}

}

void pop(){ max=max-1;

cout<<"Number is: "<<head->data<<endl; head=head->next;

}

bool isEmpty(){ if(head==NULL) return 1; else return 0;

}

bool isFull(){ if(max>=9) return 1; else return 0;

}

};

int main(){ int ch,c=1; queues s; while(c==1){ cout<<"1. Insert"<<endl; cout<<"2. Remove"<<endl; cout<<"3. Is Full"<<endl; cout<<"4. Is Empty"<<endl; cout<<"Choice: "; cin>>ch; switch(ch){ case 1:

if(s.isFull()) cout<<"Stack is full"<<endl; else

s.push(); break; case 2:

if(s.isEmpty())

cout<<"Stack is empty"<<endl;

else

s.pop(); break; case 3:

if(s.isFull()) cout<<"Queue is full"<<endl;

else

cout<<"Queue is not full"<<endl; break; case 4:

if(s.isEmpty()) cout<<"Queue is empty"<<endl;

else

cout<<"Queue is not Empty"<<endl; break; default:

cout<<"Invalid Choice"<<endl;

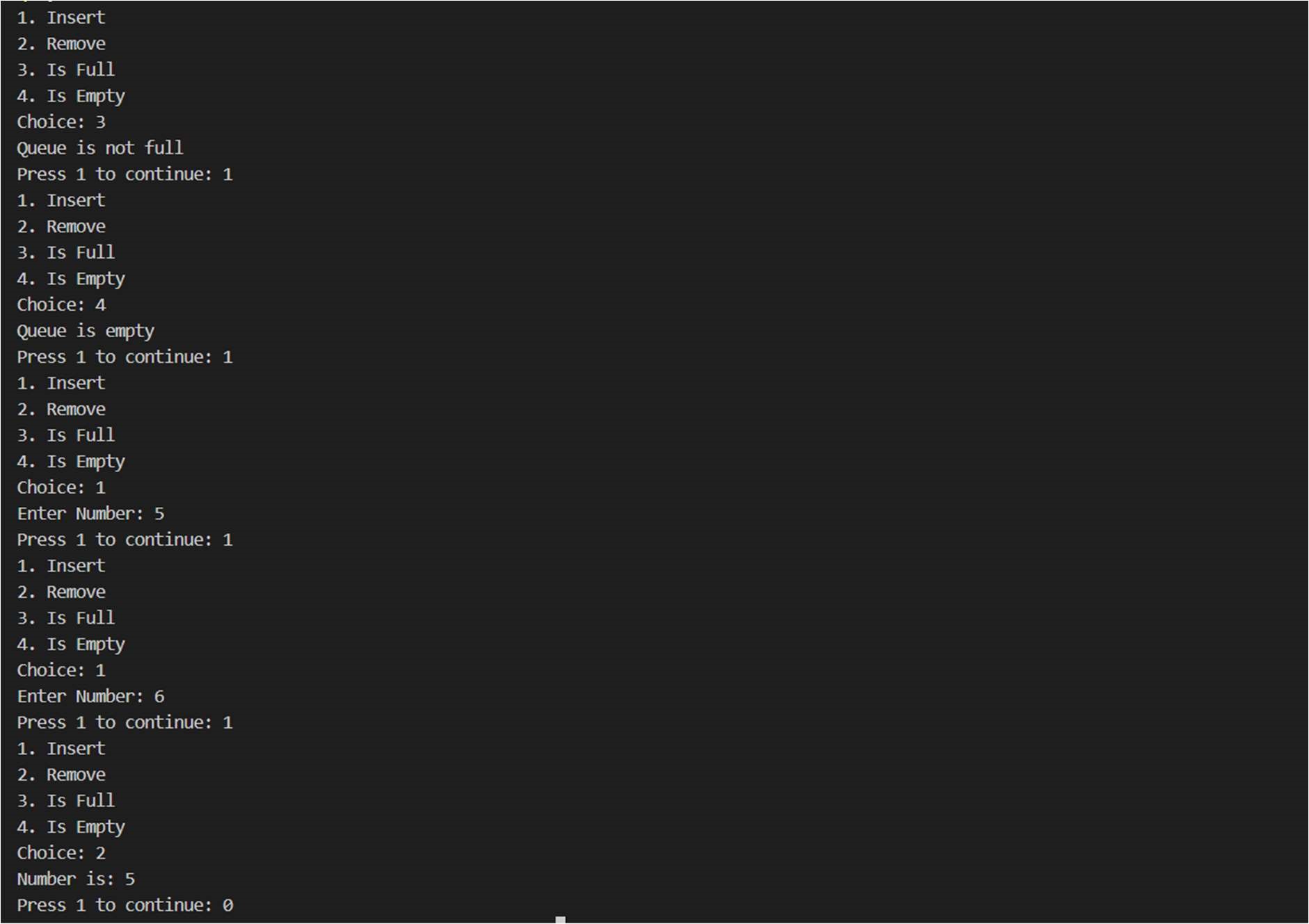
}

cout<<"Press 1 to continue: "; cin>>c;

}

return 0;

}



Q4. Implement a queue using two stacks only. Name the two as forward and reverse stacks.

The resultant queue (comprised of two stacks) should perform the following functions with minimum time complexity: i. Enqueue() ii. Dequeue()

1. isFull()
2. isEmpty()

Also incorporate the code for exception handling.

#include <iostream> using namespace std; class queues{ private:

int arr1[10]; int arr2[10]; int top=-1,top1=0; public:

void enqueue(int num){ top=top+1; arr1[top]=num;

}

void dequeue(){

cout<<"Number is: "<<arr1[0]<<endl;

if(top==0) top=-1; else{ top1=0;

for(int i=top;i>0;i--,top1=top1+1) arr2[top1]=arr1[i]; top=-1; for(int i=top1-1;i>-1;i--){ top=top+1; arr1[top]=arr2[i];

}

}

}

bool isFull(){ if(top==9) return 1; else return 0;

}

bool isEmpty(){ if(top==-1) return 1; else return 0;

}

};

int main(){ int ch,c=1,num,pos; queues s; while(c==1){ cout<<"1. Enqueue"<<endl; cout<<"2. Dequeue"<<endl; cout<<"3. Is Full"<<endl; cout<<"4. Is Empty"<<endl; cout<<"Choice: "; cin>>ch; switch(ch){ case 1:

if(s.isFull()) cout<<"Queue is full"<<endl;

else{

cout<<"Enter Number: "; cin>>num;

s.enqueue(num);

} break; case 2:

if(s.isEmpty()) cout<<"Queue is empty"<<endl;

else

s.dequeue(); break;

case 3:

if(s.isFull()) cout<<"Queue is full"<<endl;

else

cout<<"Queue is not full"<<endl; break; case 4:

if(s.isEmpty()) cout<<"Queue is empty"<<endl;

else

cout<<"Queue is not Empty"<<endl;

break; default:

cout<<"Invalid Choice"<<endl;

}

cout<<"Press 1 to continue: "; cin>>c;

}

return 0;

}



Q5. Implement priority queue by writing a class and also implement the following functions:

* Insert or enqueue
* Remove or dequeue
* Isfull
* Isempty

Write a main function to exemplify the results. Also write a main function to make the implementation a “Menu-Driven”.

#include <iostream> using namespace std; class prqueue{ private:

int data[10]; int pri[10]; int top=-1,max; public:

void insert(int num,int pr){ top=top+1; data[top]=num; pri[top]=pr;

}

void remove(){

max=0;

for(int i=1;i<=top;i++){ if(pri[i]>pri[max]) max=i;

}

cout<<"Number is: "<<data[max]<<" with priority: "<<pri[max]<<endl; for(int i=max;i<top;i++){ data[i]=data[i+1]; pri[i]=pri[i+1];

}

top=top-1;

}

bool isFull(){ if(top==9) return 1; else return 0;

}

bool isEmpty(){ if(top==-1) return 1; else return 0;

}

};

int main(){

int ch,c=1,num,pr; prqueue s; while(c==1){ cout<<"1. Insert"<<endl; cout<<"2. Delete"<<endl; cout<<"3. Is Full"<<endl; cout<<"4. Is Empty"<<endl; cout<<"Choice: "; cin>>ch; switch(ch){ case 1:

if(s.isFull()) cout<<"Stack is full"<<endl; else{

cout<<"Enter Number: "; cin>>num; cout<<"Enter Priority: "; cin>>pr;

s.insert(num,pr);

} break; case 2:

if(s.isEmpty()) cout<<"Queue is empty"<<endl; else

s.remove();

break; case 3:

if(s.isFull()) cout<<"Queue is full"<<endl;

else

cout<<"Queue is not full"<<endl; break; case 4:

if(s.isEmpty()) cout<<"Queue is empty"<<endl;

else

cout<<"Queue is not Empty"<<endl;

break; default:

cout<<"Invalid Choice"<<endl;

}

cout<<"Press 1 to continue: "; cin>>c;

}

return 0;

}

