|  |  |
| --- | --- |
| **Shoubra Faculty of Engineering**  **Benha University** | **Computer Engineering Department**  **3rd year** |

**Data Structure: Assignment #6**

**Programming problems:**

1. Implement a **Deque** data structure using a linked list (the **Deque** should be able to work as a Stack or as a Queue) for simplicity, implement it using a double linked list. You need to implement four methods:

void **pushBack**( *SomeType* data)

void **pushFront**(*SomeType* data)

*SomeType* **popBack**()

*SomeType* **popFront**().

2. Write and test this method: *PriorityQ* **merge (***PriorityQ* **q1,** *PriorityQ* **q2)** the method merges two priority queues into a single one and returns this new one.

3. Write a method *Queue* **reverse** (*Queue* q) for reversing the order of a queue.

**Hint:** You are freely to use another data structure (if you need) to do your task.

**Assignment Problem:**

- From your study about Queue data structure, Design, implement and test a **circular priority queue**.

**Question 1:**

package que;

public class node {

int number;

node next,prev;

public node(){

next=null;

prev=null;

}

public node(int x){

next=null;

prev=null;

number=x;

}

void print (){

System.out.println(number);

}

}

public class qq {

node front;

node end;

public qq(){

front=null;

end=null;

}

boolean isempty(){

return front==null;

}

void pushback(int n){

node x=new node();

x.number=n;

if(front==null){

front=x;

end=x;

}

else{

end.next=x;

x.prev=end;

end=x;}

}

void pushfront(int n){

node x=new node();

x.number=n;

if(front==null){

front=x;

end=x;

}

else{

x.next=front;

front.prev=x;

front=x;

}

}

node popback(){

node y=end;

end=end.prev;

return y;

}

node popfront(){

node y=front;

front=front.next;

return y;

}

}

public class Que {

public static void main(String[] args) {

qq testq=new qq();

testq.pushfront(1);

testq.pushfront(3);

testq.pushfront(9);

for(int i=0;i<3;i++)

testq.popback().print();

}

}

**Question 2**

package pque;

class PriorityQ

{

// array in sorted order, from max at 0 to min at size-1

private int maxSize;

private long[] queArray;

int nItems;

//-------------------------------------------------------------

public PriorityQ(int s) // constructor

{

maxSize = s;

queArray = new long[maxSize];

nItems = 0;

}

public void insert(long item) // insert item

{

int j;

if(nItems==0) // if no items,

queArray[nItems++] = item; // insert at 0

else // if items,

{

for(j=nItems-1; j>=0; j--) // start at end,

{

if( item > queArray[j] ) // if new item larger,

queArray[j+1] = queArray[j]; // shift upward

else // if smaller,

break; // done shifting

} // end for

queArray[j+1] = item; // insert it

nItems++;

} // end else (nItems > 0)

} // end insert()

public long remove() // remove minimum item

{ return queArray[--nItems]; }

//-------------------------------------------------------------

public long peekMin() // peek at minimum item

{ return queArray[nItems-1]; }

//-------------------------------------------------------------

public boolean isEmpty() // true if queue is empty

{ return (nItems==0); }

//-------------------------------------------------------------

public boolean isFull() // true if queue is full

{ return (nItems == maxSize); }

//-------------------------------------------------------------

} // end class PriorityQ

public class Pque {

public static void main(String[] args) {

// TODO code application logic here

PriorityQ pq1=new PriorityQ(7);

PriorityQ pq2=new PriorityQ(9);

pq1.insert(1) ;

pq1.insert(72) ;

pq1.insert(14) ;

pq1.insert(8) ;

pq2.insert(93) ;

pq2.insert(3) ;

pq2.insert(75) ;

pq2.insert(12) ;

pq2.insert(22) ;

pq2.insert(2) ;

PriorityQ M=merge(pq1,pq2);

int y=M.nItems;

for (int i=0;i<y;i++){

System.out.println(M.remove());

}

}

public static PriorityQ merge (PriorityQ q1, PriorityQ q2){

int x=q1.nItems;

int y=q2.nItems;

int z=x+y;

PriorityQ qm=new PriorityQ(z);

for(int i=0; i<x;i++){

qm.insert(q1.remove());

}

for(int i=0; i<y;i++){

qm.insert(q2.remove());

}

return qm;

}

}