Lab 6

Priority Queues and Lists stored as Arrays

1. Questions. (10 points)

1.1 What values are stored in each entry for a priority queue? (Choose two)  
  
Index

Key

Next

Previous

Superordinate

Value

1.2. How many items can you put on a queue?  
  
As many as you allocate when you create the queue  
  
As many as will fit in the computer's memory  
  
As many as will fit in a single string  
  
Ten

1.3. Which operations does a priority queue support? (Choose two)  
  
insert(k, v) removeMax() removeMin() add(k, v)

1.4 Please complete the following table.

|  |  |  |
| --- | --- | --- |
| Operation | Priority Queue | Output |
| Insert(3, “Apple”); | (3,”Apple”) |  |
| Insert(5, “Windows”); | (3,”Apple”), (5, “Windows”) |  |
| removeMin(); | (5, “Windows”) | (3, “Apple”) |
| Insert(1, “Linux”); | (1, “Linux”), (5, “Windows”) |  |
| Insert(2, “BSD”); | (1, “Linux”), (2, “BSD”), (5, “Windows”) |  |
| removeMin(); | (2, “BSD”), (5, “Windows”) | (1, “Linux”) |
|  | (5, “Windows”) |  |
| Insert(7, “RiscOS”); | (5, “Windows”), (7, “RiscOS”) |  |

1.7. Please complete the following table.

|  |  |  |
| --- | --- | --- |
| Operation | Priority Queue | Output |
| Insert(5, “A”) | (5, “A”) | - |
| removeMin() | - | (5, “A”) |
| insert(2, “B”) | (2, “B”) | - |
| Insert(4, “D”) | (2, “B”), (4, “D”) | - |
| removeMin() | (4, “D”) | (2, “B”) |
| Insert(1, “A”) | (1, “A”), (4, “D”) | - |
| Insert (5, “E”) | (1, “A”), (4, “D”), (5, “E”) | - |
| removeMin() | (4, “D”), (5, “E”) | (1, “A”) |

Rubric:

1.1 - 1.3: 1 points per question

1.7: 3 points

1.8: 4 points

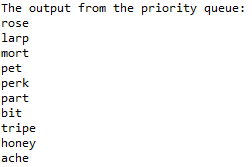
2. Priority queues in Java. (10 points)

A sample file of work with Java's Priority Queue class is provided with the lab. Adapt the provided program to do the following:

2.1. Input a String value from the user.  
2.2. Input an Integer key from the user.  
2.3. Repeat #2.1 and 2.2 for a total of **ten** items input.

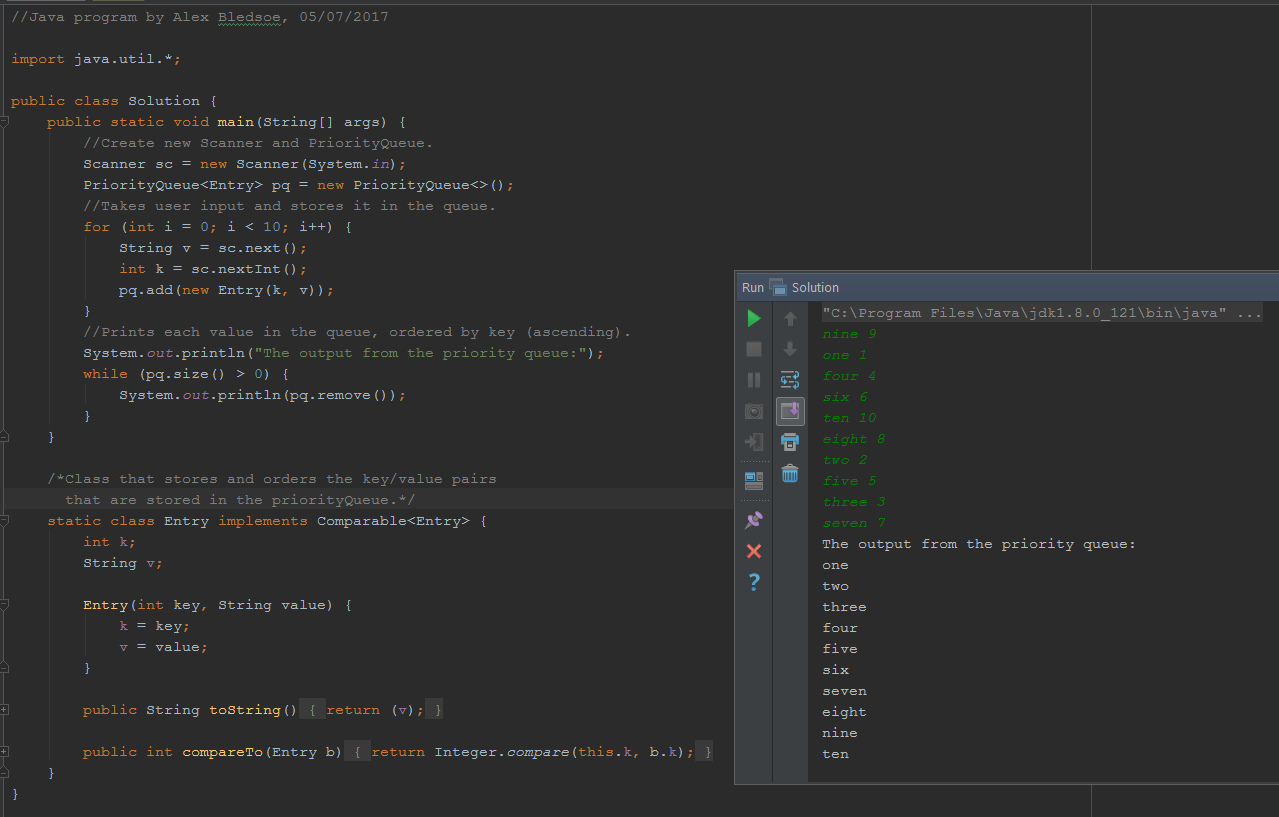
2.4. Use pq.removeMin() to display the **values** sorted by key order. Your program should not display the keys, only the values.

Sample program input and output follows. Your program must work for all reasonable user input, not just the rest case.



Rubric:  
Student name and today’s date is a comment in the first line of the programs: -5 points if fails  
Screenshot and program code: -5 points if fails   
Input from the user: 2 points  
String -> int conversion: 2 points  
Priority Queue populated with 10x(key, value): 4 points  
Output is values only, in key order: 2 points

Please paste a screenshot of a successful program run, and copy-and-paste the source code from your main program's .java file, here. You do not need to include the textbook PriorityQueue.java file.



**Solution.java:**

//Java program by Alex Bledsoe, 05/07/2017  
  
import java.util.\*;  
  
public class Solution {  
 public static void main(String[] args) {  
 //Create new Scanner and PriorityQueue.  
 Scanner sc = new Scanner(System.*in*);  
 PriorityQueue<Entry> pq = new PriorityQueue<>();  
 //Takes user input and stores it in the queue.  
 for (int i = 0; i < 10; i++) {  
 String v = sc.next();  
 int k = sc.nextInt();  
 pq.add(new Entry(k, v));  
 }  
 //Prints each value in the queue, ordered by key (ascending).  
 System.*out*.println("The output from the priority queue:");  
 while (pq.size() > 0) {  
 System.*out*.println(pq.remove());  
 }  
 }  
  
 /\*Class that stores and orders the key/value pairs  
 that are stored in the priorityQueue.\*/  
 static class Entry implements Comparable<Entry> {  
 int k;  
 String v;  
  
 Entry(int key, String value) {  
 k = key;  
 v = value;  
 }  
  
 public String toString() {  
 return (v);  
 }  
  
 public int compareTo(Entry b) {  
 return Integer.*compare*(this.k, b.k);  
 }  
 }  
}

3. Lists stored in arrays (5 points)

3.1. Which ADT operation corresponds to A[2] = 5?

setA(2,5) set(A, 2, 5) set(5,2,A) A.set(2,5)

3.2. Which ADT operation corresponds to x = A[3]?

setX(A,3) x = A.get(3) A.get(3) = x x.get(3) = A

3.3. In your own words: if a list is stored as an array, why is adding an item in the middle of the list slow?

**- Because all the elements that come after the element that’s being added must be moved to the next subsequent index.**

3.4. In your own words: if a list is stored as an array, why is removing an item from the middle of the list slow?

**- Because all the elements in the list that came after the element that’s being removed must be shifted to the next earliest index.**

3.5. If a list is stored as an array, and the list grows bigger than the array size, how might the program respond? (Choose four)

Throw an exception

Create a new array twice as big as the old one and copy the values over

Create a new array one hundred elements larger than the old one and copy the values over

Return an error

ALL

~~4. Queues in Java (10 points) Thanks to Reges Building Java Programs 4th edition.~~

~~Write a method called doubleQueue that accepts a queue and replaces every element of the queue with two copies of that element. You are welcome to use java.util.Queue.~~

~~Please program three test cases into your program. Show the queues before and after transformation:~~

~~[ 1, 2, 3 ] becomes [1,1,2,2,3,3,]~~

~~[ “a”, “b”, “c” ] becomes [ “a”, “a”, “b”, “b”, “c”, “c” ]~~

~~[ 5, 6, 7, 8, 9] becomes [5, 5, 6, 6, 7, 7, 8, 8, 9, 9]~~

~~However, your doubleQueue routine must work for~~ *~~any~~* ~~queue data, not just the test cases.~~

~~Rubric:~~

~~Student name and today’s date are a comment on the first line of the program: (-5 if fails)~~

~~Screenshot and program code: (-5 points if fails)~~

~~doubleQueue algorithm: 5 points  
Main program with test cases: 5 points~~

~~Please paste a screenshot of a successful program run, and copy-and-paste the source code from your main program's .java file, here.~~

**\*Duplicate from last lab.**

5. Comprehensive. (20 points)

Implement a priority queue using java.util.Stack as the base data type. You may need several Java classes for this: one to define Entry, one for the priority queue, and one for the main program. It's acceptable use inner class if you'd rather have everyone contained in one .java file.

Feel free to reuse code from the sample files, on-line sources, this lab, and/or the textbook. Please cite any code found on-line.

Your priority queue must implement the following methods:

insert(Entry E);

removeMin(); - returns the entry with the lowest key value

Where Entry is:

int k;

String v;

The core data storage should be:

Stack<Element> s;

Hint: you may need more than one stack to fully implement the queue!

The following test cases should be hard-coded into your program:

|  |  |
| --- | --- |
| Input (in order) | Output (in order) |
| (3, "test")  (4, "every")  (5, "edge") | (3, "test")  (4, "every")  (5, "edge") |
| (10, "chipped")  (5, "quality")  (12, "beef")  (1, "good") | (1, "good")  (5, "quality")  (10, "chipped")  (12, "beef") |
| (8, "must")  (3, "all")  (7, "things")  (10, "end") | (3, "all")  (7, "things")  (8, "must")  (10, "end") |

I am not picky about output formatting, but the data must come from the priority queue.

The program must work for any correct sequence of entries, not just the test cases.

Rubric:

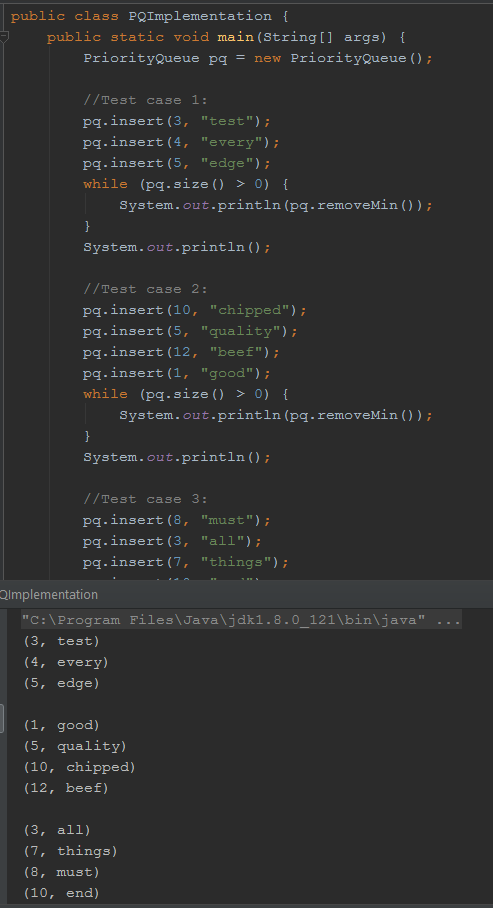
Student name and today’s date are a comment on the first line of the program: (-5 if fails)  
Screenshot and program code: (-5 if fails)

Works for other values, not just test cases: (-5 if fails)

Priority queue as stack (or two stacks): 10 points  
Test cases included in program: 5 points

Operation is correct: 5 points

Please paste a screenshot of a successful program run, and copy-and-paste the source code from your .java file(s) here:



**Entry.java:**

public class Entry {  
 private int k;  
 private String v;  
  
 Entry(int key, String value) {  
 k = key;  
 v = value;  
 }  
  
 int getKey() {  
 return k;  
 }  
  
 public String toString() {  
 return ("(" + k + ", " + v + ")");  
 }  
}

**PriorityQueue.java:**

import java.util.EmptyStackException;  
import java.util.Stack;  
  
class PriorityQueue {  
 private Stack<Entry> main;  
 private int stackSize;  
  
 //Constructor  
 PriorityQueue() {  
 main = new Stack<>();  
 stackSize = 0;  
 }  
  
 //returns the current size of the priority queue.  
 int size() {  
 return stackSize;  
 }  
  
 //returns entry with smallest key (highest priority) and removes it from the priority queue.  
 String removeMin() {  
 stackSize--;  
 return main.pop().toString();  
 }  
  
 //Inserts new entry into the priority queue based on the given key value.  
 void insert(int key, String value) {  
 Entry newest = new Entry(key, value);  
 if (main.size() == 0) {  
 main.push(newest);  
 stackSize++;  
 } else {  
 Stack<Entry> holder = new Stack<>();  
 try {  
 while (newest.getKey() >= main.peek().getKey()) {  
 holder.push(main.pop());  
 stackSize--;  
 }  
 } catch (EmptyStackException e) {  
 } finally {  
 main.push(newest);  
 stackSize++;  
 }  
 while (holder.size() > 0) {  
 main.push(holder.pop());  
 stackSize++;  
 }  
 }  
 }  
}

**PQImplementation.java:**

public class PQImplementation {  
 public static void main(String[] args) {  
 PriorityQueue pq = new PriorityQueue();  
  
 //Test case 1:  
 pq.insert(3, "test");  
 pq.insert(4, "every");  
 pq.insert(5, "edge");  
 while (pq.size() > 0) {  
 System.*out*.println(pq.removeMin());  
 }  
 System.*out*.println();  
  
 //Test case 2:  
 pq.insert(10, "chipped");  
 pq.insert(5, "quality");  
 pq.insert(12, "beef");  
 pq.insert(1, "good");  
 while (pq.size() > 0) {  
 System.*out*.println(pq.removeMin());  
 }  
 System.*out*.println();  
  
 //Test case 3:  
 pq.insert(8, "must");  
 pq.insert(3, "all");  
 pq.insert(7, "things");  
 pq.insert(10, "end");  
 while (pq.size() > 0) {  
 System.*out*.println(pq.removeMin());  
 }  
 }  
}