|  |  |  |
| --- | --- | --- |
| Sukkur_IBA_New_Logo | **Sukkur IBA University**  **Department of Computer Science** | **C:\Users\Saif Hassan\Downloads\CS logo (3).jpg** |

**DATA STRUCTURES**

**Lab07 – Complete Binary Tree , Priority Queues**

**Instructor: Saif Hassan**

**READ IT FIRST**

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don’t waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

**KEEP IT WITH YOU!**

1. Indent your code inside the classes and functions. It’s a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don’t jump to the formula directly.

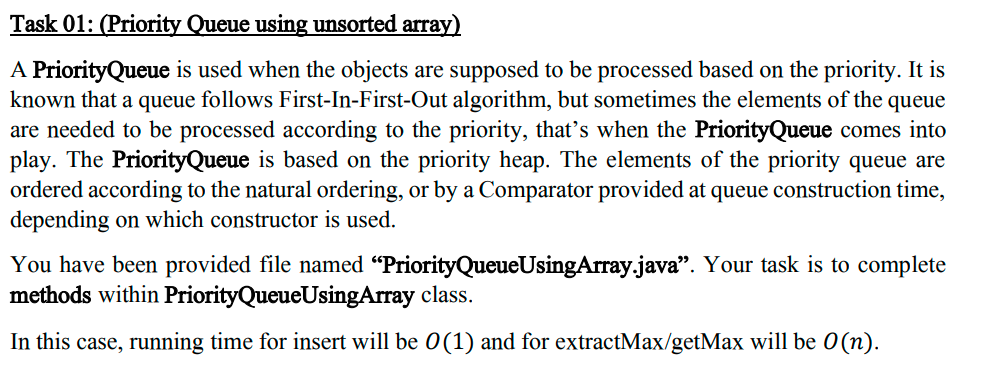
I, \_\_**Amjad Ali**\_\_\_\_\_ with student ID \_191-21-0001\_\_

Section \_\_**A**\_\_hereby declare that I do understand the instructions above and follow them. This is

my own work.

**Exercises**

**Task1 Description**



Solution:

1. import java.util.Arrays;
3. public class PriorityQueueUsingArray {
4. //unsorted Array
5. private int[] array;
6. private int rear = -1;
7. private int size;
9. public PriorityQueueUsingArray(int size) {
10. this.size = size;
11. array = new int[size];
12. }
14. public void insert(int item) {
15. if (rear + 1 == size) {
16. System.out.println("Queue is overflow");
17. } else {
18. array[++rear] = item;
19. System.out.println("Successfully added!");
20. }
22. }
24. public int getMax() {
25. if (rear == -1) {
26. System.out.println("Queue is Underflow");
27. return -1;
28. }
29. int max = 0;
30. for (int i = 0; i <= rear; i++) {
31. max = Math.max(max, array[i]);
32. }
34. return max;
36. }
38. public int extractMax() {
39. if (rear == -1) {
40. System.out.println("Queue is Underflow");
41. return -1;
42. }
43. int max = 0;
44. int ind = 0;
45. for (int i = 0; i <= rear; i++) {
46. if (max < array[i]) {
47. max = array[i];
48. ind = i;
49. }
50. }
51. for (int i = ind; i < rear; i++)
52. array[i] = array[i + 1];
54. array[rear] = 0;
55. rear--;

58. return max;
59. }
61. public boolean search(int data) {
62. if (rear == -1) {
63. System.out.println("Queue is Underflow");
64. return false;
65. }
67. for (int i = 0; i <= rear; i++) {
68. if (data == array[i]) {
69. return true;
70. }
71. }
73. return false;

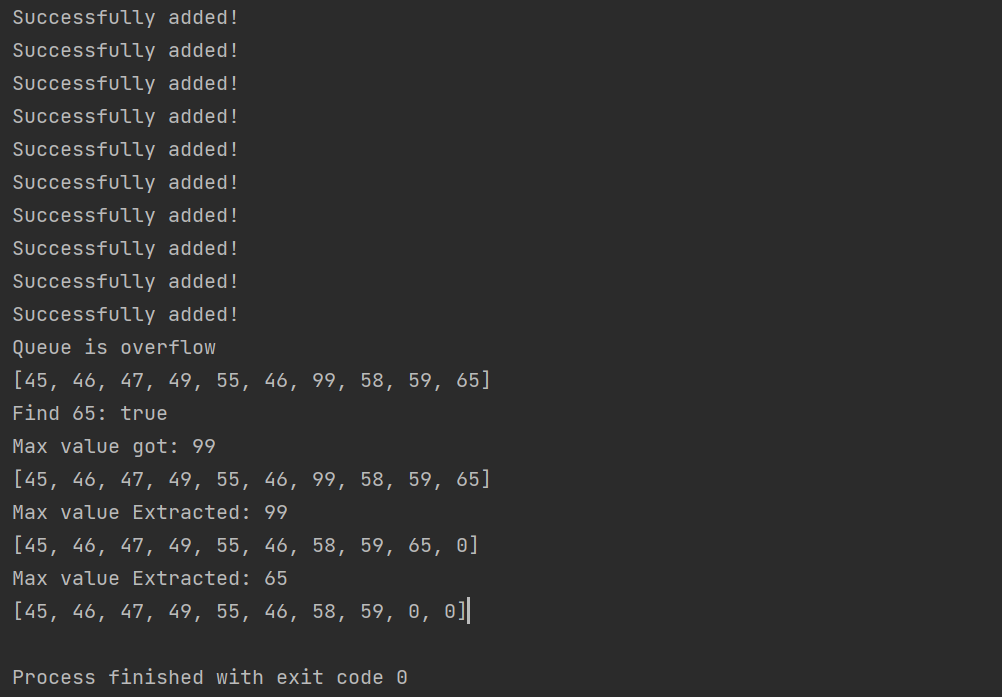
76. }
78. public static void main(String[] args) {
80. PriorityQueueUsingArray q = new PriorityQueueUsingArray(10);
81. q.insert(45);
82. q.insert(46);
83. q.insert(47);
84. q.insert(49);
85. q.insert(55);
86. q.insert(46);
87. q.insert(99);
88. q.insert(58);
89. q.insert(59);
90. q.insert(65);
91. q.insert(66);
92. System.out.println(Arrays.toString(q.array));
93. System.out.println("Find 65: " + q.search(65));
94. System.out.println("Max value got: " + q.getMax());
95. System.out.println(Arrays.toString(q.array));
96. System.out.println("Max value Extracted: " + q.extractMax());
97. System.out.println(Arrays.toString(q.array));
98. System.out.println("Max value Extracted: " + q.extractMax());
99. System.out.println(Arrays.toString(q.array));

102. }
104. }

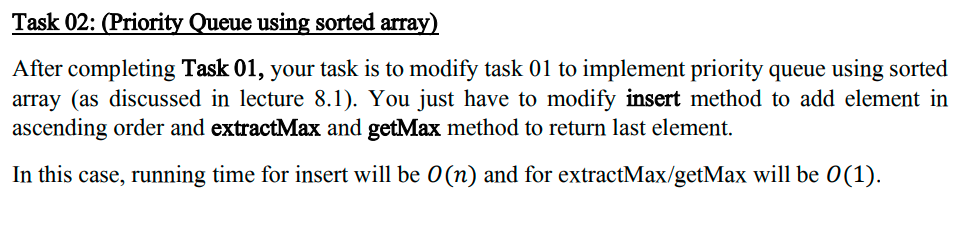
**Sample Input:**

****

**Sample Output**

****

**Task2 Description**



Solution:

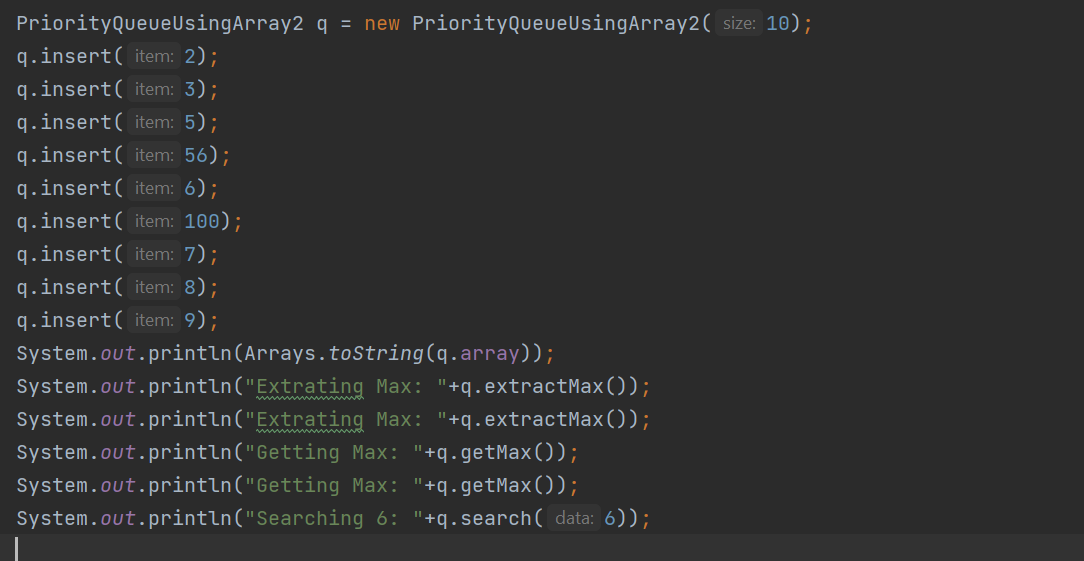
1. import java.util.Arrays;
2. import java.util.LinkedList;
3. import java.util.Queue;
5. public class PriorityQueueUsingArray2 {
6. //sorted Array
7. private int[] array;
8. private int rear = -1;
9. private int size;
11. public PriorityQueueUsingArray2(int size) {
12. this.size = size;
13. array = new int[size];
14. }
16. public void insert(int item) {
17. if (rear + 1 == size) {
18. System.out.println("Queue is overflow");
19. } else {
20. if (rear == -1) {
21. array[++rear] = item;
22. return;
23. }
25. boolean condition = false;
26. int temp = 0;
28. for (int i = 0; i <= rear; i++) {

31. if (!(item > array[i]) && !condition) {
32. temp = array[i];
33. array[i] = item;
34. condition = true;
35. continue;
36. }
38. if (condition) {
40. int temp2 = array[i];
41. array[i] = temp;
42. temp = temp2;
43. }
44. }
45. if (!condition) {
46. array[++rear] = item;
47. } else {
48. array[++rear] = temp;
49. }
50. System.out.println("Successfully added!");
51. }
53. }
55. public int getMax() {
56. if (rear == -1) {
57. System.out.println("Queue is Underflow");
58. return -1;
59. }
60. return array[rear];
62. }
64. public int extractMax() {
65. if (rear == -1) {
66. System.out.println("Queue is Underflow");
67. return -1;
68. }
69. int temp = array[rear];
70. array[rear] = 0;
71. rear--;
72. return temp;
73. }
75. public boolean search(int data) {
76. if (rear == -1) {
77. System.out.println("Queue is Underflow");
78. return false;
79. }
81. for (int i = 0; i <= rear; i++) {
82. if (data == array[i]) {
83. return true;
84. }
85. }
87. return false;

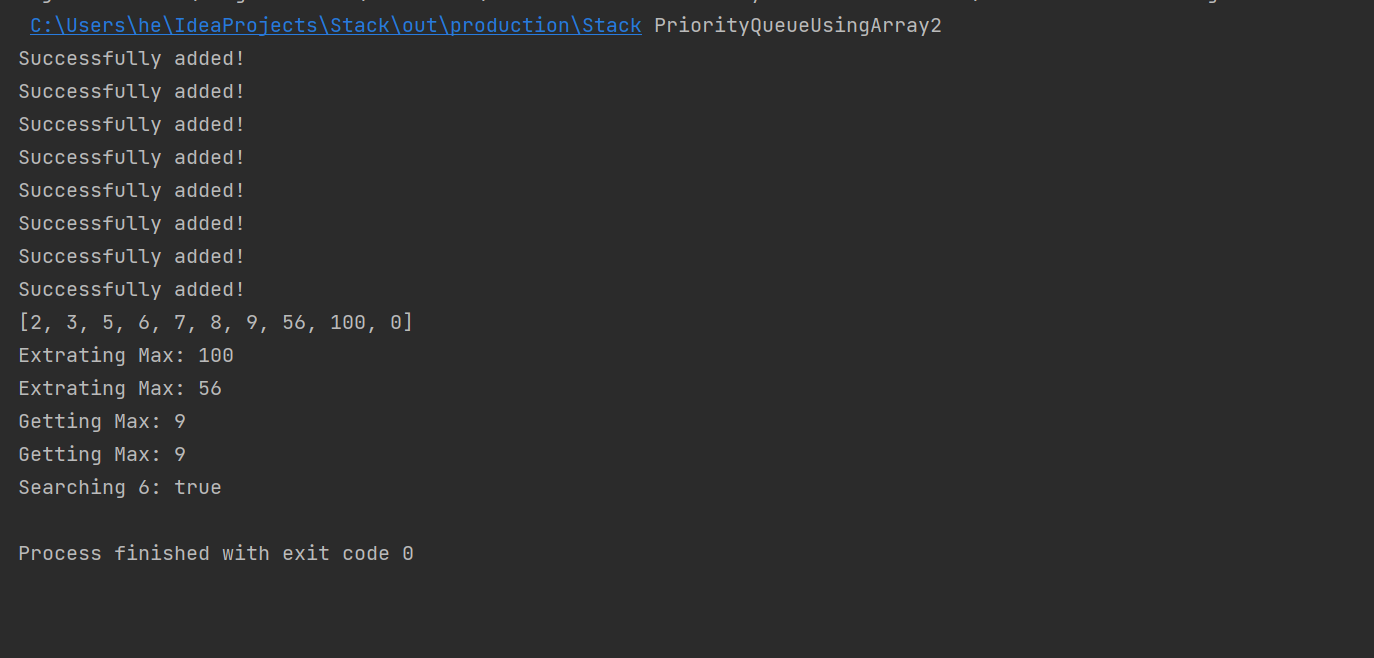
90. }
92. public static void main(String[] args) {
94. PriorityQueueUsingArray2 q = new PriorityQueueUsingArray2(10);
95. q.insert(2);
96. q.insert(3);
97. q.insert(5);
98. q.insert(56);
99. q.insert(6);
100. q.insert(100);
101. q.insert(7);
102. q.insert(8);
103. q.insert(9);
104. System.out.println(Arrays.toString(q.array));
105. System.out.println("Extrating Max: " + q.extractMax());
106. System.out.println("Extrating Max: " + q.extractMax());
107. System.out.println("Getting Max: " + q.getMax());
108. System.out.println("Getting Max: " + q.getMax());
109. System.out.println("Searching 6: " + q.search(6));

112. }
113. }

**Sample Input:**

****

**Sample Output**



**Task3 Description**

**…**

**…**

**…**

**…**

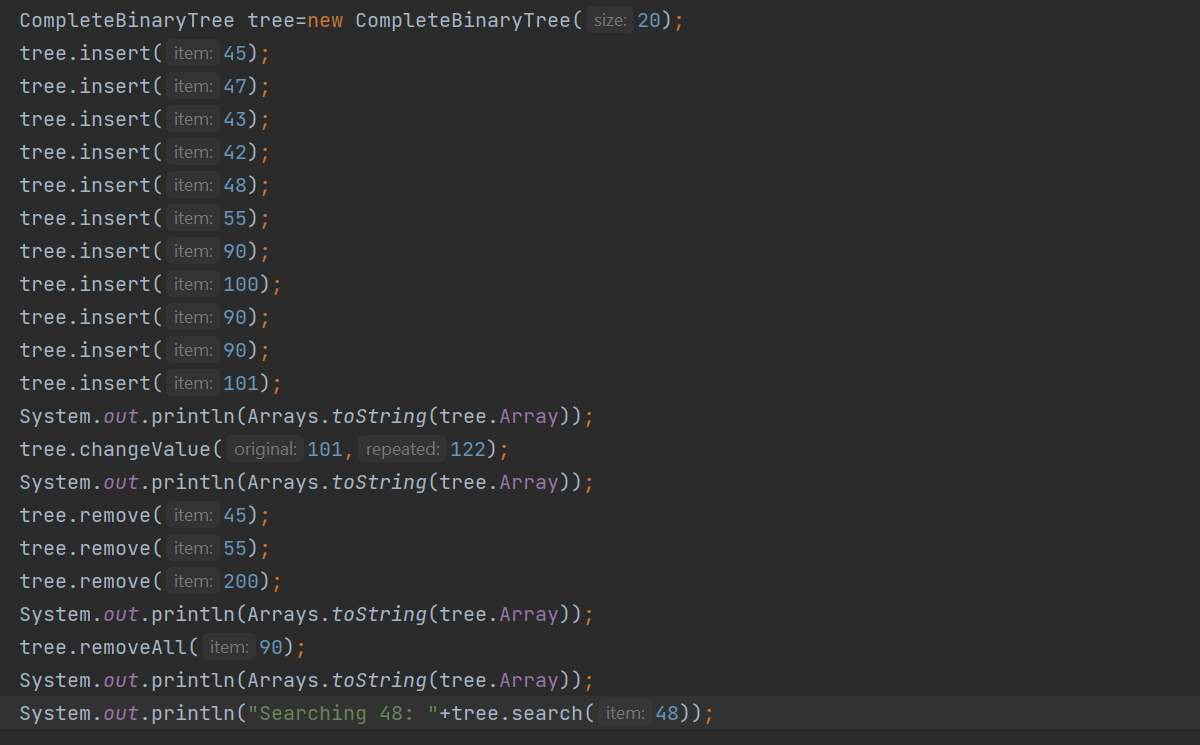
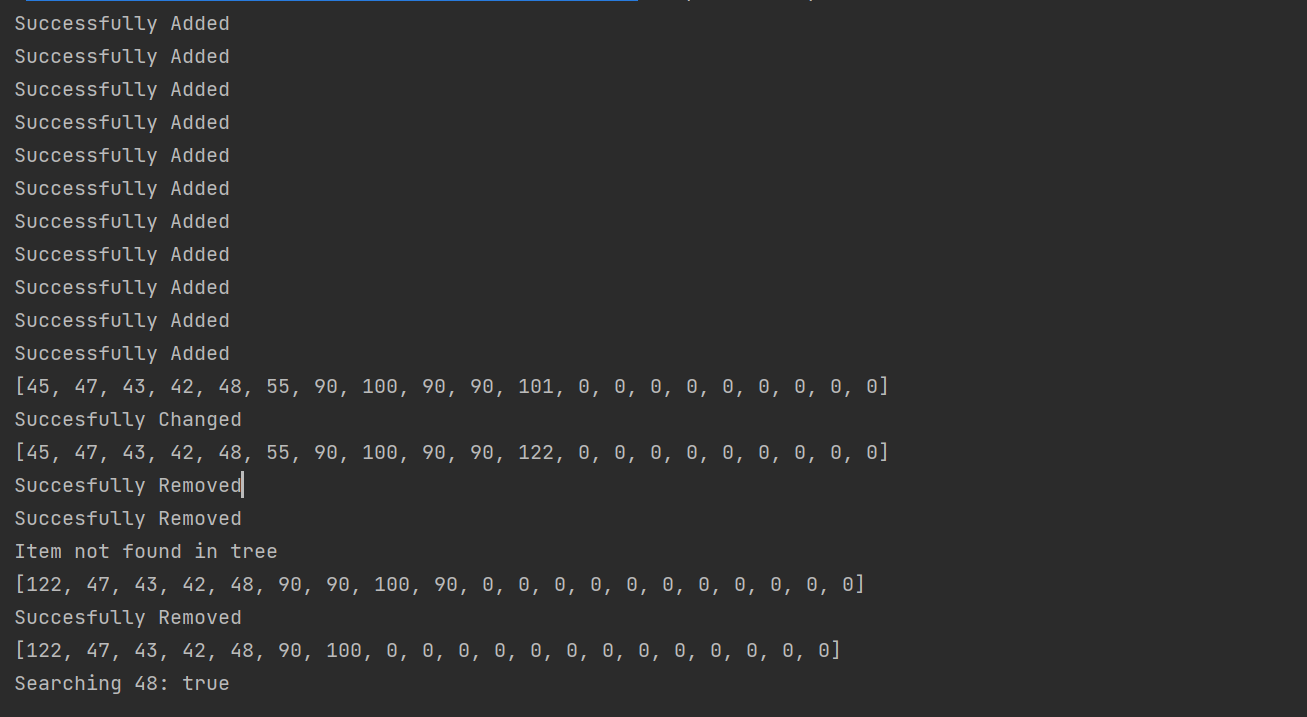
Solution:

1. import java.util.Arrays;
3. public class CompleteBinaryTree {
4. int[] Array;
5. int size;
6. int rear = -1;
8. CompleteBinaryTree(int size) {
9. this.size = size;
10. Array = new int[size];
11. }
13. public void insert(int item) {
14. if (rear + 1 == size) {
15. System.out.println("Tree is Overflow");
16. return;
17. }
19. Array[++rear] = item;
20. System.out.println("Successfully Added");
21. }
23. public void remove(int item) {
24. if (rear == -1) {
25. System.out.println("Tree is underflow");
26. return;
27. }
28. boolean cond = false;
29. for (int i = 0; i <= rear; i++) {
30. if (Array[i] == item) {
31. cond = true;
32. Array[i] = Array[rear];
33. Array[rear] = 0;
34. rear--;
35. break;
36. }
37. }
39. if (cond)
40. System.out.println("Succesfully Removed");
41. else
42. System.out.println("Item not found in tree");
43. }
45. public void removeAll(int item) {
46. if (rear == -1) {
47. System.out.println("Tree is underflow");
48. return;
49. }
50. boolean cond = false;
51. for (int i = 0; i <= rear; i++) {
52. if (Array[i] == item) {
53. cond = true;
54. Array[i] = Array[rear];
55. Array[rear] = 0;
56. rear--;
57. }
58. }
60. if (cond)
61. System.out.println("Succesfully Removed");
62. else
63. System.out.println("Item not found in tree");
64. }
66. public void changeValue(int original, int repeated) {
67. if (rear == -1) {
68. System.out.println("Tree is underflow");
69. return;
70. }
71. boolean cond = false;
72. for (int i = 0; i <= rear; i++) {
73. if (Array[i] == original) {
74. cond = true;
75. Array[i] = repeated;
76. }
77. }
79. if (cond)
80. System.out.println("Succesfully Changed");
81. else
82. System.out.println("Item not found in tree");
83. }
85. public boolean search(int item) {
86. if (rear == -1) {
87. System.out.println("Tree is underflow");
88. return false;
89. }
90. for (int i = 0; i <= rear; i++) {
91. if (Array[i] == item) {
92. return true;
93. }
94. }
96. return false;
98. }
100. public static void main(String[] args) {
102. CompleteBinaryTree tree = new CompleteBinaryTree(20);
103. tree.insert(45);
104. tree.insert(47);
105. tree.insert(43);
106. tree.insert(42);
107. tree.insert(48);
108. tree.insert(55);
109. tree.insert(90);
110. tree.insert(100);
111. tree.insert(90);
112. tree.insert(90);
113. tree.insert(101);
114. System.out.println(Arrays.toString(tree.Array));
115. tree.changeValue(101, 122);
116. System.out.println(Arrays.toString(tree.Array));
117. tree.remove(45);
118. tree.remove(55);
119. tree.remove(200);
120. System.out.println(Arrays.toString(tree.Array));
121. tree.removeAll(90);
122. System.out.println(Arrays.toString(tree.Array));
123. System.out.println("Searching 48: " + tree.search(48));

126. }

129. }

**Sample Input:**

**Sample Output**