

Homework Turnin

Account: 6G_06 (rgalanos@fcps.edu)
Section: 6G
Course: TJHSST APCS 2016-17
Assignment: 06-07
Receipt ID: cb803e22be7dafcc9a61e4d01f5a98ef

Turnin Successful!

The following file(s) were received:

McRonal3.java (5650 bytes)

```
1. //name :   date:
2.
3. import java.util.*;
4. public class McRonal3
5. {
6.     public static final int TIME = 1079; //18 hrs * 60 min
7.     public static void main(String[] args)
8.     {
9.         int numberCustomers = 0;
10.        int totalWait = 0;
11.        int longestWait = 0;
12.        int currentSize = 0;
13.        int longestQueue = 0;
14.        int[] servingTime = {100, 100, 100};
15.        int[] count = {0, 0, 0};
16.
17.        Queue<Integer> customers = new LinkedList<Integer>();
18.        Queue<Integer> service1 = new LinkedList<Integer>();
19.        Queue<Integer> service2 = new LinkedList<Integer>();
20.        Queue<Integer> service3 = new LinkedList<Integer>();
21.
22.        for(int i=0; i<TIME; i++)
23.        {
24.            if(Math.random()<0.5)
25.            {
26.                customers.add(new Integer(i));
27.                currentSize++;
28.                numberCustomers++;
29.                if(currentSize>longestQueue)
30.                    longestQueue = currentSize;
31.            }
32.
33.            if(!service1.isEmpty())
34.            {
35.                count[0]++;
36.                if(servingTime[0]==count[0])
37.                {
38.                    int x = i - service1.remove();
39.                    totalWait+=x;
40.                    if(x>longestWait)
41.                        longestWait = x;
42.                }
43.            }
44.            if(!service2.isEmpty())
45.            {
46.                count[1]++;
47.                if(servingTime[1]==count[1])
48.                {
49.                    int x = i - service2.remove();
50.                    totalWait+=x;
```

```

51.         if(x>longestWait)
52.             longestWait = x;
53.     }
54. }
55. if(!service3.isEmpty())
56. {
57.     count[2]++;
58.     if(servingTime[2]==count[2])
59.     {
60.         int x = i - service3.remove();
61.         totalWait+=x;
62.         if(x>longestWait)
63.             longestWait = x;
64.     }
65. }
66.
67.
68. if(service1.isEmpty()&&!customers.isEmpty())
69. {
70.     service1.add(customers.remove());
71.     servingTime[0] = (int)(Math.random()*6+2);
72.     count[0] = 0;
73.     currentSize--;
74. }
75. if(service2.isEmpty()&&!customers.isEmpty())
76. {
77.     service2.add(customers.remove());
78.     servingTime[1] = (int)(Math.random()*6+2);
79.     count[1] = 0;
80.     currentSize--;
81. }
82. if(service3.isEmpty()&&!customers.isEmpty())
83. {
84.     service3.add(customers.remove());
85.     servingTime[2] = (int)(Math.random()*6+2);
86.     count[2] = 0;
87.     currentSize--;
88. }
89.
90. System.out.print(i+": ");
91. // display(customers);
92. display(merge(service1, service2, service3, customers));
93. }
94.
95. int counter = 1079;
96. while(!(customers.isEmpty()&&service1.isEmpty()&&service2.isEmpty()&&service3.isEmpty()))
97. {
98.     if(!service1.isEmpty())
99.     {
100.        count[0]++;
101.        if(servingTime[0]==count[0])
102.        {
103.            int x = counter - service1.remove();
104.            totalWait+=x;
105.            if(x>longestWait)
106.                longestWait = x;
107.        }
108.    }
109.    if(!service2.isEmpty())
110.    {
111.        count[1]++;
112.        if(servingTime[1]==count[1])
113.        {
114.            int x = counter - service2.remove();
115.            totalWait+=x;
116.            if(x>longestWait)
117.                longestWait = x;
118.        }
119.    }
120.    if(!service3.isEmpty())
121.    {
122.        count[2]++;
123.        if(servingTime[2]==count[2])
124.        {
125.            int x = counter - service3.remove();
126.            totalWait+=x;
127.            if(x>longestWait)
128.                longestWait = x;
129.        }
130.    }
131. }

```

```

132.
133.     if(service1.isEmpty()&&!customers.isEmpty())
134.     {
135.         service1.add(customers.remove());
136.         servingTime[0] = (int)(Math.random()*6+2);
137.         count[0] = 0;
138.         currentSize--;
139.     }
140.     if(service2.isEmpty()&&!customers.isEmpty())
141.     {
142.         service2.add(customers.remove());
143.         servingTime[1] = (int)(Math.random()*6+2);
144.         count[1] = 0;
145.         currentSize--;
146.     }
147.     if(service3.isEmpty()&&!customers.isEmpty())
148.     {
149.         service3.add(customers.remove());
150.         servingTime[2] = (int)(Math.random()*6+2);
151.         count[2] = 0;
152.         currentSize--;
153.     }
154.
155.     System.out.print(counter+": ");
156.     // display(customers);
157.     display(merge(service1, service2, service3, customers));
158.     counter++;
159. }
160.
161. System.out.println("Total customers served = " + numberCustomers);
162. System.out.println("Average wait time = " + (double)totalWait/numberCustomers);
163. System.out.println("Longest wait time = " + longestWait);
164. System.out.println("Longest queue = " + longestQueue);
165. }
166. public static Queue<Integer> merge(Queue<Integer> a, Queue<Integer> b, Queue<Integer> c, Queue<Integer> d)
167. {
168.     Queue<Integer> temp = new LinkedList<Integer>();
169.     Queue<Integer> temp1 = new LinkedList(a);
170.     Queue<Integer> temp2= new LinkedList(b);
171.     Queue<Integer> temp3= new LinkedList(c);
172.     Queue<Integer> temp4 =new LinkedList(d);
173.
174.
175.     while(!temp1.isEmpty())
176.         temp.add(temp1.remove());
177.     while(!temp2.isEmpty())
178.         temp.add(temp2.remove());
179.     while(!temp3.isEmpty())
180.         temp.add(temp3.remove());
181.     while(!temp4.isEmpty())
182.         temp.add(temp4.remove());
183.
184.     return temp;
185. }
186. public static void display(Queue<Integer> q)
187. {
188.     System.out.println(q);
189. }
190. }
191.

```