Seoul National University

M1522.000900 Data Structure

Spring 2025, Kang

Homework 2: Lists, Stacks, and Queues & Binary Trees

(Chapters 4 & 5)

Due: **23:59, April 29th (Tuesday), 2025**

**Reminders**

* Lead TA: Jeongin Yun (snuds.ta@gmail.com)
* The points of this homework add up to 100.
* All assignments must be done individually.
* **Type your answer in English** and submit your assignment **in PDF format**.
  + Answer written in Korean may get 0 points.
  + You will get a 10% deduction on your grade for this homework if your submission is either hand-written or non-PDF format.
* Name your file as “(studentID)-(name)-HW2.pdf”.  
  (e.g., *202512345-GildongHong-HW2.pdf*)
* Whenever you are making an assumption, state it clearly.
* If you have any questions about the assignment, post them on eTL.

**Submission**

* Submit your assignment to **eTL**.
* The submission after the due date will be regarded as a late **submission, even if it is only one second late.** Late submissions are accepted **within only one week after the due date.**
* You do not need to specify whether to use the slip-days; they are automatically used.

**Question 1 [18 points]**

The following table is a circular array-based queue. This queue has a front and rear variable. Perform the following enqueue and dequeue operations in order on the queue starting from the initial queue (e.g. perform operation (1) on the initial queue, operation (2) on the resulting queue from part (1), etc.). Place the result below the operation and mark where the front (F) and rear (R) pointer are at the end of the operation. Refer to the class implementation of the queue in **Figure 1**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Initial queue]   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Index** | 0 | 1 | 2 | 3 | 4 | | **Front/Rear** |  |  | F | R |  | | **Initial Queue** |  |  | z | z |  | | |
| (1) enqueue(‘q’) [3 points]   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Index** | 0 | 1 | 2 | 3 | 4 | | **Front/Rear** |  |  |  |  |  | | **Queue** |  |  |  |  |  | | (2) dequeue() [3 points]   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Index** | 0 | 1 | 2 | 3 | 4 | | **Front/Rear** |  |  |  |  |  | | **Queue** |  |  |  |  |  | |
| (3) enqueue(‘n’) [3 points]   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Index** | 0 | 1 | 2 | 3 | 4 | | **Front/Rear** |  |  |  |  |  | | **Queue** |  |  |  |  |  | | (4) dequeue() [3 points]   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Index** | 0 | 1 | 2 | 3 | 4 | | **Front/Rear** |  |  |  |  |  | | **Queue** |  |  |  |  |  | |
| (5) enqueue(‘x’) [3 points]   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Index** | 0 | 1 | 2 | 3 | 4 | | **Front/Rear** |  |  |  |  |  | | **Queue** |  |  |  |  |  | | (6) enqueue(‘f’) [3 points]   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Index** | 0 | 1 | 2 | 3 | 4 | | **Front/Rear** |  |  |  |  |  | | **Queue** |  |  |  |  |  | |

|  |
| --- |
| public class Queue<E> {  private int size = 4;  private int maxSize = size+1;  private int front = 1;  private int rear = 0;  private E[] listArray = (E[]) new Object[maxSize];  public void enqueue(E it) {  assert ((rear + 2) % maxSize) != front : "Queue is full";  rear = (rear + 1) % maxSize;  listArray[rear] = it;  }  public E dequeue() {  assert length() != 0 : "Queue is empty";  E it = listArray[front];  front = (front + 1) % maxSize;  return it;  }  public int length() {  return (rear + maxSize - front + 1) % maxSize;  } |

**Figure 1.** An array-based queue implementation

**Question 2 [10 points]**

The following is a singly linked list’s node class and a “mystery” recursive method.

|  |
| --- |
| public class LinkedListTracing {      static class Node {          String data;          Node next;          Node(String data) {              this.data = data;              this.next = null;          }      }      public static Node mystery(Node node) {          if (node == null) {              return null;          }          node.next = mystery(node.next);          if (node.data.length() >= 7) {              node.data = node.data + "1332";          }          if (node.data.charAt(0) != 'A') {              return node;          } else {              return node.next;          }      }  } |

You are given the following linked list:

Elena

Brandon

Austin

Destini

nulli

head

Draw the linked list after the following code is executed:

|  |
| --- |
| mystery(head); |

**Question 3 [10 points]**

Show the change in stack S resulting from the following functions of **Stack ADT (Figure 2)**:

S.push(10);

S.push(5);

S.pop();

S.push(7);

S.pop();

S.push(4);

Assume that the stack is empty at the beginning of the series, but it is not initialized after each function is executed. For each function, you should represent the resulting stack status in text, following the format in the right column of the example below. [20 points]

|  |  |
| --- | --- |
| **Example:** S.push(1); | |
| C:\Users\전현식\Downloads\20170930215127.png | [ / / / 1 (top) ] |

|  |
| --- |
| ​​/\*\* Stack ADT \*/  public interface Stack {     /\*\* Reinitialize the stack.\*/     public void clear();     /\*\* Push an element onto the top of the stack.\*/     public void push(E it);     /\*\* Remove and return the element at the top of the stack.\*/     public E pop();      /\*\* @return A copy of the top element. \*/     public E topValue();     /\*\* @return The number of elements in the stack. \*/      public int length();}; |

**Figure 2.** The stack ADT

Your answer:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.push(10); | S.push(5); | S.pop(); | S.push(7); | S.pop(); | S.push(4); |
|  |  |  |  |  |  |

**Question 4 [20 points]**

Postfix expression is in the form of “A-B-O”. A and B are numbers or also postfix expressions, and O is an operator like ‘+’, ‘-’, ‘\*’, ‘/’. For example, a commonly used infix expression “1 + 2” is “1 2 +” in the postfix expression. Answer the following questions.

(1) Transform the following infix expression to a postfix expression. [5 points]

(2 + 5) \* 3 \* (4 +7)

(2) Assume that there is a calculator that takes a postfix expression as an input and calculates the result. When the calculator reads a digit number, it is pushed into the stack. When it reads an operator, it pops two values from the stack, calculates the operation with the two values, and pushes the result to the stack. Show the change in stack resulting from the following postfix expression (each element in the expression is split by space):

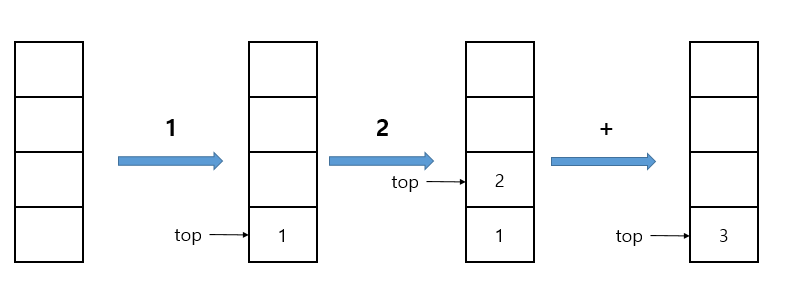
**[input expression]**

3 7 9 \* + 10 5 / -

For each element of expression, represent the resulting stack status in text. Below is an example. [15 points]

**[Example expression]**

1 2 +



**[Example answer]**

1. [ / / / ]
2. [ / / / 1 (top) ]
3. [ / / 2 (top) / 1 ]
4. [ / / / 3 (top) ]

**Question 5 [10 points]**

Suppose that we have binary search trees which have integers as node values, and we want to search for the number 365. Answer whether the following sequences could be or could not be the sequence of nodes examined. Write the brief reason for invalid sequences.

(1) 2, 252, 401, 398, 330, 344, 397, 365 [2 points]

(2) 924, 220, 911, 244, 898, 258, 362, 365 [2 points]

(3) 925, 202, 911, 240, 912, 245, 365 [2 points]

(4) 2, 399, 387, 219, 266, 382, 381, 278, 365 [2 points]

(5) 935, 278, 347, 621, 299, 392, 358, 365 [2 points]

**Question 6 [12 points]**

A max-heap is a heap where the value of each parent is greater than or equal to the values of its children. Answer the following questions.

스케치, 그림, 도표, 원이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.스케치, 그림, 도표, 라인 아트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.(1) Which of the following binary trees is a max-heap? You can choose multiple trees. If it's not a max-heap, provide a brief explanation. [6 points]

**Figure 3. Binary trees**

(2) Where in a max-heap might the smallest element reside, assuming that all elements are distinct? [2 points]

(3) What is the time complexity of the following operations in a max-heap? Use Big-Oh notation. [4 points]

|  |  |
| --- | --- |
| Operation | Time complexity |
| Insert |  |
| Remove |  |
| Max search |  |
| Min search |  |

**Question 7 [20 points]**

For this question only, handwriting is permitted. However, if the handwriting is unclear or illegible to the grader, points will be deducted.

(1) Construct a Binary Search Tree by inserting the following integers in the given order. [4 points]

|  |
| --- |
| 50, 30, 70, 20, 40, 65, 85, 10, 25 |

(2) Insert the following additional values into the tree from part (1) in the given order. [4 points]

|  |
| --- |
| 15, 27, 35, 90 |

(3) Now delete the following nodes from the updated tree in this exact order.

\* Deletion Rule: If the node to be deleted has two children, always replace its value with the inorder successor. The inorder successor is the node that would be printed immediately after the current node in an inorder traversal. [4 points]

|  |
| --- |
| 25, 30, 65 |

(4) For the resulting tree from part (3), list the node values in each of the following traversals. [4 points]

(i) Inorder  
(ii) Preorder  
(iii) Postorder

(5) What is the **height** of the tree from part (3)? Which node(s) have **maximum depth** in that tree? [4 points]