CSCI2100D 2023-24: Solution of Assignment 2[*]

[#] This assignment is due at 11:59:59pm, 6th March 2024.

- **Q1. [16 marks]** Answer the following questions about Stacks.
  - **(i). [8 marks]** Given the postfix expression 2 6 - 5 * 1 4 + /, show how to use a stack to calculate the final results. Please show the stack status step by step (Hint. You may follow the steps as shown in `CSCI2100D-Lectures-9-Stack-Queue`).

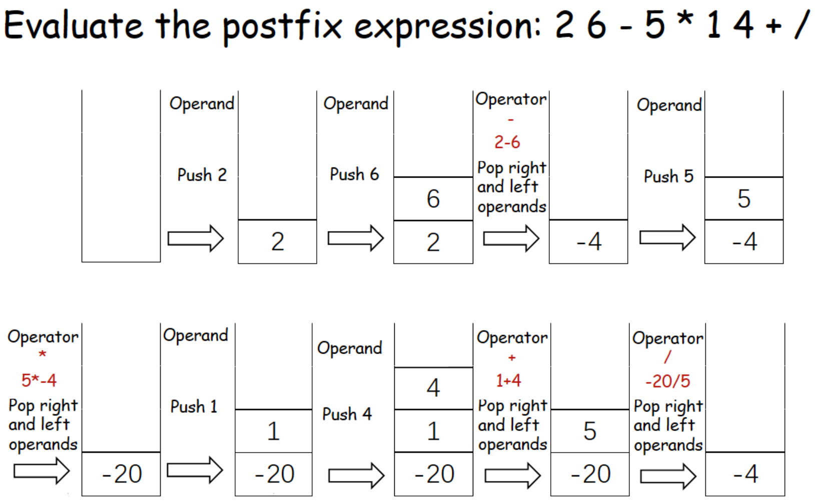
Evaluate the postfix expression: 2 6 - 5 * 1 4 + /
Operand
Push 2
2
Operand
Push 6
6
2
Operator
-
2-6
Pop right and left operands
-4
Operand
Push 5
5
-4
Operator
*
5*-4
Pop right and left operands
-20
Operand
Push 1
1
-20
Operand
Push 4
4
1
-20
Operator
+
1+4
Pop right and left operands
5
-20
Operator
/
-20/5
Pop right and left operands
-4

Figure 1. Solution of Q1(i)

  As shown in Figure 1, we return -4 as the answer.
  - **(ii). [8 marks]** Use a stack to check if the symbol list [{ ( ) }] [( ]) is balanced. Show the stack status after each symbol checking (Hint. You may follow the steps as shown in `CSCI2100D-Lectures-9-Stack-Queue`).
  As shown in Figure 2, we find a mismatch when checking the penultimate item. Therefore, return false.

[*]Departmental Guideline for Plagiarism (Department of Systems Engineering and Engineering Management): If a student is found plagiarizing, his/her case will be reported to the Department Examination Panel. If the case is proven after deliberation, the student will automatically fail the course in which he/she committed plagiarism. The definition of plagiarism includes copying of the whole or parts of written assignments, programming exercises, reports, quiz papers, mid-term examinations and final examinations. The penalty will apply to both the one who copies the work and the one whose work is being copied, unless the latter can prove his/her work has been copied unwittingly. Furthermore, inclusion of others' works or results without citation in assignments and reports is also regarded as plagiarism with similar penalty to the offender. A student caught plagiarizing during tests or examinations will be reported to the Faculty office and appropriate disciplinary authorities for further action, in addition to failing the course.

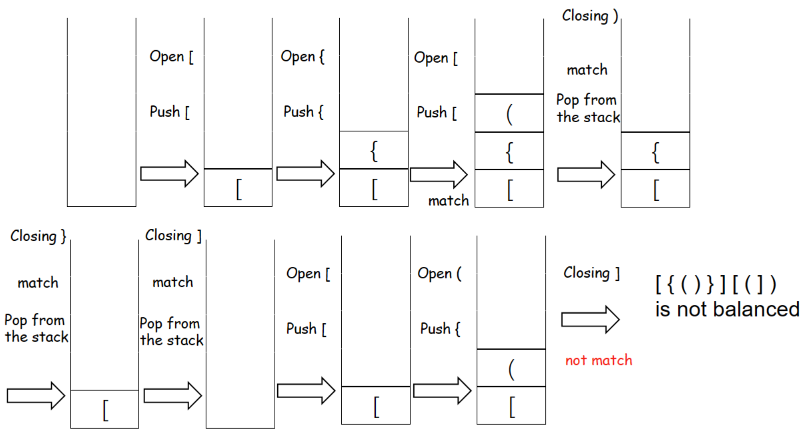

Figure 2. Solution of Q1 (ii)

- **Q2. [10 marks]** Answer the following questions about Queues.
  - **(i). [5 marks]** Assume that we have an empty queue Q. Given a series of queue operations on Q as below, write down the element returned by each Dequeue operation and show the final queue.
    - Enqueue(Q, 1), Enqueue(Q, 3), Dequeue(Q), Enqueue(Q, 2), Enqueue(Q, 5), Dequeue(Q), Dequeue(Q), Enqueue(Q, 4), Dequeue(Q).

    The item returned are 1, 3, 2, and 5 in order. The final queue contains only 4.
  - **(ii). [5 marks]** Suppose that we use a circular queue, what is the minimum size required to support the above series of queue operations? If the value of front is initially set to 0, what will the final values of the front and rear be?

    After executing Enqueue(Q, 5), the number of elements in the queue reaches its maximum value of 3. Therefore, the array size we need is 4.
    Finally, the front is 0 and the rear is 1.
- **Q3. [30 marks]** Answer the following questions about Trees.
  - **(i) [7 marks]** Suppose we have a binary tree T_1. The inorder traversal sequence and postorder traversal sequence of T_1 are given below. Please draw the tree out and also give the preorder traversal sequence of T_1.
    - Inorder traversal sequence of T_1: d, f, a, b, c, g, e
    - Postorder traversal sequence of T_1: d, a, f, c, e, g, b

    From the Postorder sequence we know that root data is b. Then with root data b and Inorder sequence, we can easily see that d, f, a belong to the left child tree and c, g, e belong to the right child tree. Then we go back to Postorder sequence. f is the last one of three left data items in Postorder, which means f is the root of left

child tree. Similarly, we can see that g is the root data of right child tree. Then the structure of the entire tree can be drew out easily, as shown in Figure 3.
Preorder traversal sequence of T: b, f, d, a, g, c, e.

- **(ii). [8 marks]** Given a binary tree T_2 as shown in Fig. 1, is T_2 a max heap? Justify your answer. Next, write down the array representation of the binary tree T_2. Please fill the values in the array as shown below.

60	52	42	50	14	12	38	46	18				
[1]	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**[10]**	**[11]**	**[12]**	**[13]**

Answer: T_2 is a max heap. First, T_2 is a complete binary tree as shown in the above array. Then, for node 60, its children 52 and 42 are less than itself; for node 52, its children 50 and 14 are less than itself; for node 42, its children 12 and 38 are less than itself; for node 50, its children 46 and 18 are less than itself; other nodes are leaf nodes.

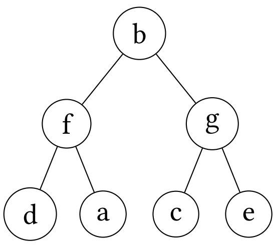

Figure 3. Solution for Q3(i)

- **(iii). [15 marks]** Design an algorithm in pseudo-code to find the maximum of **leaf nodes** in a binary tree. Your algorithm should use the following Binary Tree ADT operations. For example, the maximum of all leaf nodes of the binary tree T_2 shown in Fig. 1 is 46.
You must ensure that your algorithm will not throw ERROR in any case.

Binary Tree **ADT**

- **`Create()`:** Return an empty binary tree.
- **`MakeBT(bintreeL,element,bintreeR)`:** Return a binary tree whose left subtree is **bintreeL** and right subtree is **bintreeR**, and whose root node contains the data **element**.
- **`IsEmpty(bintree)`:** If **bintree** is empty return TRUE else return FALSE.
- **`Lchild(bintree)`:** If **bintree** is empty *throw* ERROR else return the left subtree of **bintree**.
- **`Rchild(bintree)`:** If **bintree** is empty *throw* ERROR else return the right subtree of **bintree**.
- **`Data(bintree)`:** If **bintree** is empty *throw* ERROR else return the element data stored in the root node of **bintree**.

The algorithm can be designed using recurrence as follows:

```
Algorithm MaxLeaf(bintree)
1:  if IsEmpty(bintree) return $-\infty$
2:  if IsEmpty(Lchild(bintree)) and IsEmpty(Rchild(bintree))
3:      return Data(bintree)
4:  return MAX(MaxLeaf(Lchild(bintree)),MaxLeaf(Rchild(bintree)))
```

- **Q4. [18 marks]** Answer the following questions about the Heap.
  - **(i). [9 marks]** Given the max heap *H* as shown in Fig. 2, show the procedure of inserting 47 into the max heap step by step (Hint. You may follow the steps as shown in `CSCI2100D-Lectures-11-12-BinaryTreeADT-and-Heap`).
    The insert procedure is shown in Figure 4.

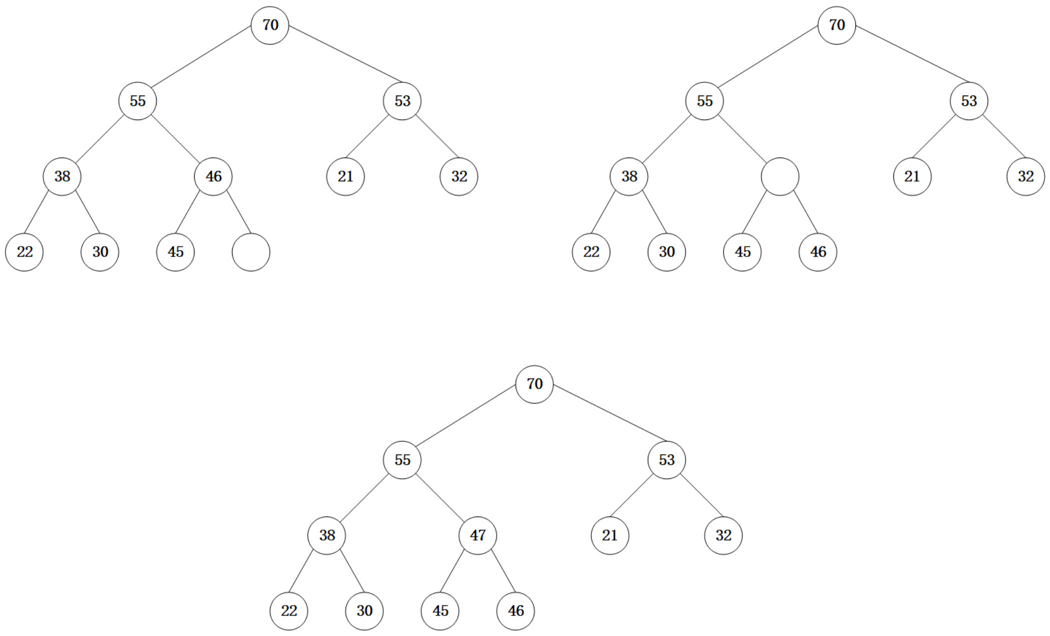

Figure 4. Solution of Q4(i)

  - **(ii). [9 marks]** Given a max heap *H* as shown in Fig. 2, show the procedure of heap delete operation on the max heap step by step (Hint. You may follow the steps as shown in `CSCI2100D-Lectures-11-12-BinaryTreeADT-and-Heap`).
    The delete procedure is shown in Figure 5.

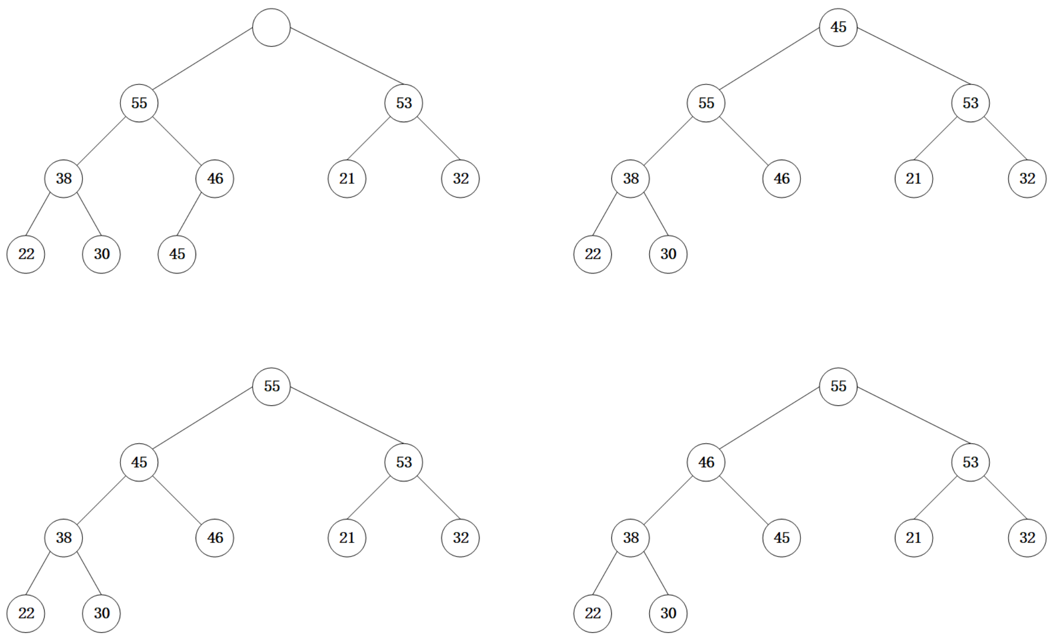

Figure 5. Solution of Q4(ii)

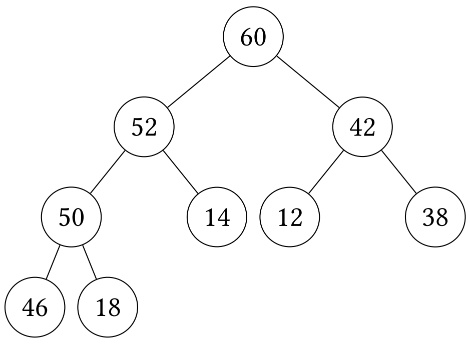

Fig. 1. A Binary Tree T_2 for Q3

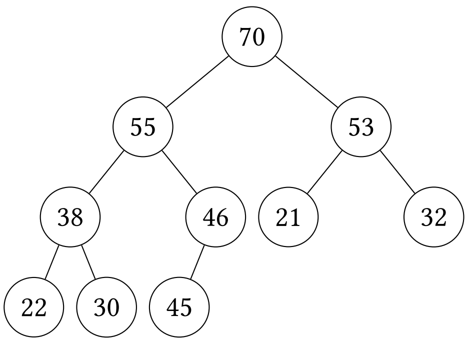

Fig. 2. A Max Heap H for Q4

- **Q5. [18 marks]** Answer the following questions about the Binary Search Tree. (Hint. You may follow the steps as shown in `CSCI2100D-Lectures-13-Binary-Search-Tree`).
  - **(i). [4 marks]** Given a binary search tree T_3 as shown in Fig. 3, show the nodes examined when searching for 85.
    The nodes examined are 81, 90, 88.
  - **(ii). [7 marks]** Given a binary search tree T_3 as shown in Fig. 3, draw the binary search tree after inserting 26, 42 in order.
    The binary tree T_3 after inserting 26, 42 is shown in Figure 6.

– **(iii). [7 marks]** Given a binary search tree T_3 as shown in Fig. 3, draw the binary search tree after deleting 81, 54 in order.
Since each deletion allows for choosing either the predecessor or successor of the deleted node to replace it, the answer is not unique. Figure 7 shows one of the possible solutions.

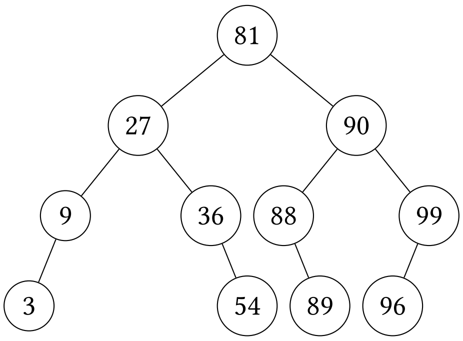

Fig. 3. A Binary Search Tree T_3 for Q5

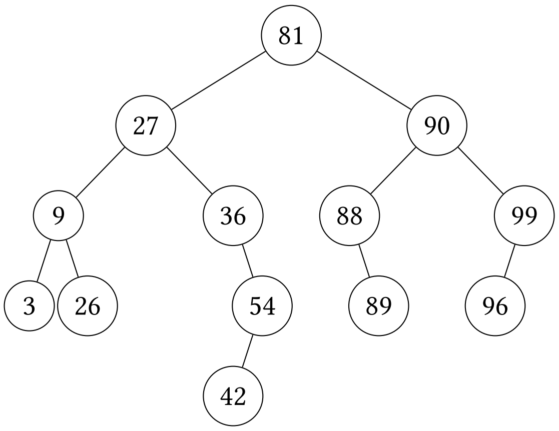

Figure 6. Solution for Q5(ii)

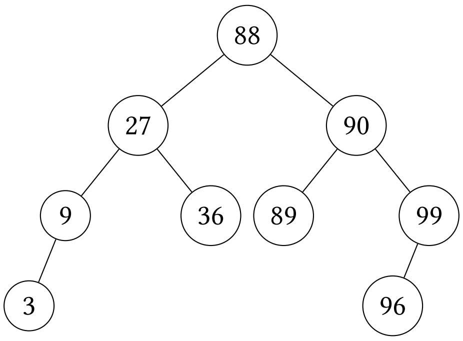

Figure 7. One solution for Q5(iii)

- **Q6. [8 marks]** In computer science, a trie is a tree data structure to locate specific keys within a set. These keys are most often strings, with links between nodes defined are individual characters. In order to access a key, the trie is traversed depth-first, following the links between nodes, which represent each character in the key.
  Though tries can be keyed by character strings, they need not be. In real-world computer architectures, integers are typically represented in binary using a fixed number of bits. For example, representing 24 in binary with 6 bits would be 011000. Therefore, we can also use a binary trie to store a set of integers. The root node represents an empty binary string. Starting from the most significant bit, each node's left child represents the next bit being 0, and its right child represents the next bit being 1 in the binary representation. Fig. 4 shows a trie T_4 which maintains a set containing 0, 3, 5, 6 represented in 3-bit binary.

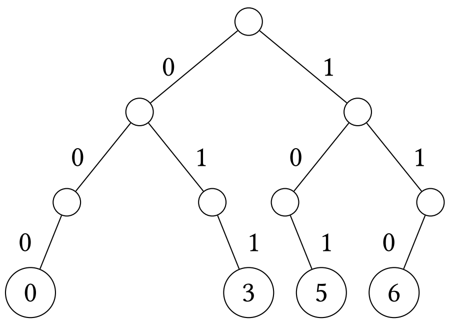

Fig. 4. A Trie T_4 maintains a set containing 0, 3, 5, 6

- **(i). [4 marks]** Draw a trie to maintain the set containing 1, 7, 12, 20, 24, 32, 63 represented in 6-bit binary. The trie is shown in Figure 8.

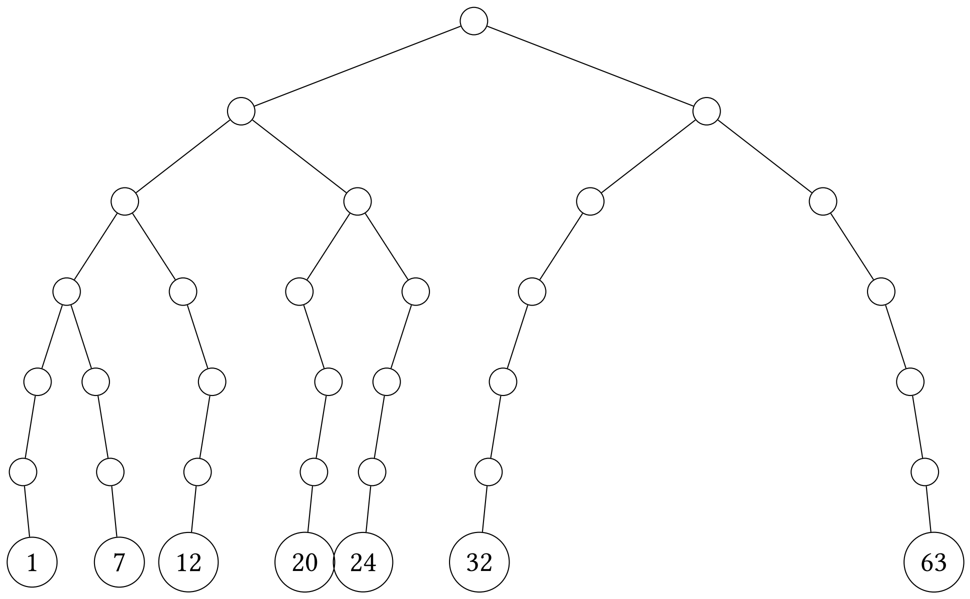

Figure 8. Solution for Q6(i)

- **(ii). [4 marks]** Suppose you have a trie maintains a set containing several integers represented in D-bit binary. Can you design an algorithm to find the largest element in this set that is not greater than the input k in O(D) time? Please introduce your design idea briefly.

  We consider different situations starting from the most significant bit to the least significant bit of k:
  - The current bit is 0.
    - If the current node has no left child, there are no element less than k in the subset.
    - Otherwise, recursively traverse to the left subtree for the next bit.
  - The current bit is 1.
    - If the current node has no right child, return the maximum element in the subset represented by its left subtree.
    - Recursively traverse to the right subtree for the next bit.
      - If we find an element in the subset represented by its right subtree, return that element.
      - Otherwise, if the current node has no left child, there are no element less than k in the subset. Else, return the maximum element in the subset represented by its left subtree.

  Note that, following this scheme, we either directly find k itself or search for the maximum element in some left subtree at most once. Based on the properties of the trie, finding the maximum element in a subtree simply requires moving as far to the right side as possible. Therefore, since the depth of all leaf nodes is D, this search process is O(D).