CSCI2100C 2019-20: Assignment 2[*]

$^{\#}$ This assignment is due at 11:59:59pm, 19th March 2020.

- **Q1. [30 marks]** Stacks and Queues.
  - **(i). [3 marks]** Assume that we have an empty stack S.
    - Given a series of stack operations on S as below:
    - Push$(S, 8)$, Push$(S, 5)$, Push$(S, 3)$, Pop(S), Pop(S), Push$(S, 7)$, and Pop(S).
    - Output the element returned by each Pop operation.
  - **(ii). [3 marks]** Assume that we have an empty queue Q.
    - Given a series of queue operations on Q as below:
    - Enqueue$(Q, 9)$, Dequeue(Q), Enqueue$(Q, 6)$, Enqueue$(Q, 3)$, Dequeue(Q), Enqueue$(Q, 4)$ and Dequeue(Q).
    - Output the element returned by each Dequeue operation.
  - **(iii). [12 marks]** Given the postfix expression 1 2 + 7 8 4 / - * 5 6 - *, 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 CSCI2100C-Lecture8-Stack-Applications Page 20).
  - **(iv). [12 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 CSCI2100C-Lecture8-Stack-Applications Page 8).
- **Q2. [28 marks]** Trees.
  - **(i). [8 marks]** Given a binary tree T as shown in Figure 1, is T a max heap? Justify your answer. Next, write down the array representation of the binary tree T. Please fill the values in the array as shown below.

[1]	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**[10]**	**[11]**	**[12]**	**[13]**	**[14]**	**[15]**

  - **(ii). [10 marks]** Given the max heap H as shown in Figure 2, show the procedure of inserting 45 into the max heap step by step (Hint. You may follow the steps as shown in CSCI2100C-Lecture10-Tree Pages 15-16).

[*] 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.

  - **(iii). [10 marks]** Given a max heap *H* as shown in Figure 2, show the procedure of heap delete operation on the max heap step by step (Hint. You may follow the steps as shown in CSCI2100C-Lecture10-Tree Page 20).

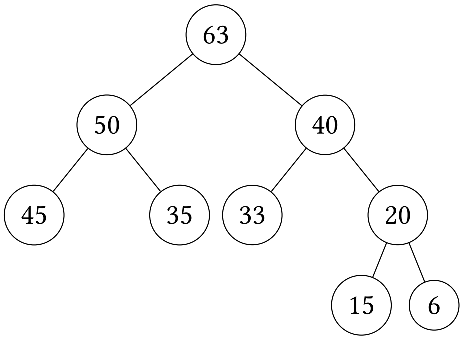

Figure 1. A Binary Tree *T* for Q2(i)

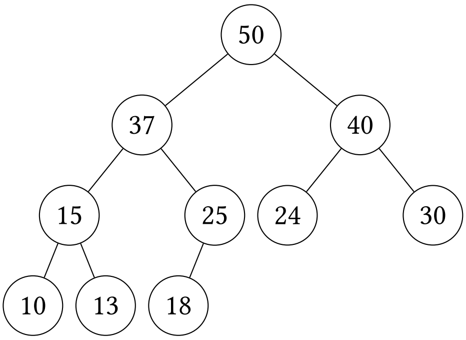

Figure 2. A Max Heap *H* for Q2(ii) and Q2(iii)

- **Q3. [24 marks]** Answer the following questions about the binary search tree.
  - **(i). [10 marks]** Given an empty binary search tree, draw the binary search tree after inserting 30, 20, 55, 60, 10, 70, 25, 80, 35, 40 in order.
  - **(ii). [2 marks]** Given a binary search tree as shown in Figure 3, which node is the successor of node 29?
  - **(iii). [2 marks]** Given a binary search tree as shown in Figure 3, which node is the predecessor of node 42?
  - **(iv). [10 marks]** Given a binary search tree as shown in Figure 3, draw the binary search tree after deleting 50, 10, 20 in order.

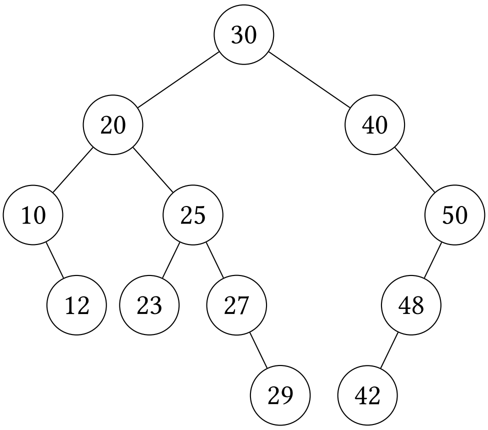

Figure 3. A Binary Search Tree for Q3

- **Q4. [18 marks]** Given a binary search tree of n nodes and its ADT defined below, answer the following questions.

Binary Search Tree **ADT**

- isEmpty(root): Determine whether or not the binary tree with node **root** as the root is an empty tree.
- leftChild(root): Return the left child of node **root**.
- rightChild(root): Return the right child of node **root**.
- parent(x): Return the parent of node **x**.
- height(x): Return the height of the (sub)tree rooted at node **x**.
- data(root): Return the data value in node **root**.
- leftSize(root): Return the number of nodes in the left subtree of node **root**.
- rightSize(root): Return the number of nodes in the right subtree of node **root**.

- **(i). [6 marks]** Show an algorithm in pseudo-code to find the maximum in the BST by using the above ADT operations.
- **(ii). [6 marks]** Show an algorithm in pseudo-code to check if a binary search tree is balanced or not by using the above ADT operations (Hint: Refer to CSCI2100C-Lecture11-12-Binary-Search-Tree Page 24 for the definition of balanced binary search tree).
- **(iii). [6 marks]** Show an algorithm in pseudo-code to find the k-th ($k \leq n$) largest element in the BST by using the above ADT operations.