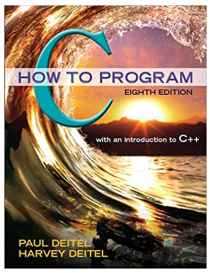
University of Windsor

Fall 2021: Final Exam
Take-Home

COMP-1410 Introduction to Algorithms & Programming II

Saturday, December 11, 2021

School of Computer Science
University of Windsor



READ THE FOLLOWING INSTRUCTION BEFORE GOING TO THE QUESTIONS

- The time limit is 3 hours.
- The exam is out of 100.
- Before starting this exam, make sure that you download the SignaturePage file, open and read it, and then type your name, initial, student id, and date into the corresponding locations, save and close the file, and include it with the files you submit on Blackboard.
- For every question:
    - You should download its corresponding C program/input files.
    - Open the program in an IDE
    - **READ THE DESCRIPTION OF THE QUESTION IN THIS DOCUMENT, AS WELL AS THE SAMPLE EXECUTION OUTPUTS.**
    - **ALSO, THOROUGHLY AND CAREFULLY READ THE COMMENTS FOR EACH FUNCTION OR PARTIAL CODE THAT YOU SHOULD COMPLETE.**
    - Complete the code.
        - You **MUST NOT ALTER/DELETE** any code that already exists inside the programs. Only add your code at the locations indicated.
    - Compile, run and test your code and compare its output with the sample execution outputs provided.

- After making sure about the correctness of your code, in terms of compile-time, run-time fatal, and run-time non-fatal errors, include the C files in your final submission.
- **FOR ANY QUESTION, IF YOUR C PROGRAM HAS COMPILE-TIME ERRORS, YOU WILL LOSE HALF OF THE ORIGINAL MARK. THEN, YOUR CODE WILL BE CHECKED LOGICALLY.**
- All the required libraries have already been included. **DO NOT include any NON-STANDARD LIBRARIES.**
- **DO NOT ALTER THE NAMES OF THE C FILES PROVIDED.**
- **DO NOT ALTER THE NAMES AND PROTOTYPES OF THE FUNCTIONS.**
- **ONLY C PROGRAMS WITH THE EXACT ORIGINAL NAMES WILL BE ACCEPTED**
- **When you finish answering the questions, SUBMIT THE C PROGRAMS, INDIVIDUALLY, on Blackboard, along with the completed SignaturePage file.**
- **No need to submit the input/output files. Only submit the source of your C programs.**
- **DO NOT CREATE ZIP FILES.**
- **DO NOT SUBMIT COMPILED FILES FOR ANY QUESTIONS. OTHERWISE, YOU WILL RECEIVE ZERO FOR THAT QUESTION.**
- **YOU SHOULD SUBMIT YOUR FILES BEFORE THE DEADLINE.**

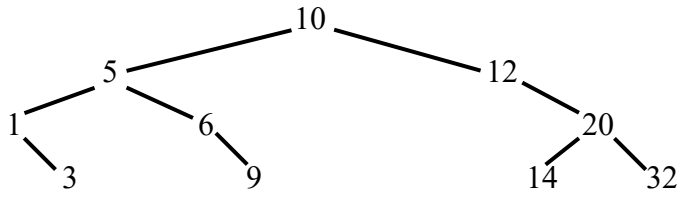
GOOD LUCK!

Question 1 (45 marks)

Binary Tree: In this question, you have been provided with an incomplete code for a Dynamic Binary Tree of integer values. Your task is to complete the code by adding some functionalities to the program. Before describing the tasks, read the following initial notes for this question:

Initial Notes:

- In this question, you have two dynamic structures, one Binary Tree, and one Queue.
  - Binary tree for the main operations in the program.
  - A Queue to use it in the function **`levelOrder`** to traverse and print the elements of the binary tree, level by level from the top (root) of the tree to the bottom (leaves).
- All the nodes' data value, which are integers, must be read from the input file, "**`input.txt`**".
- Outputs must be written to an output file, "**`output.txt`**".
- The recursive function **`insert`** for the Binary Tree, and the functions **`isQueueEmpty`**, **`enqueue`**, and **`dequeue`** for the Queue, have already been developed. **DO NOT ALTER them**. Just use them inside the other parts of the program that you must complete.
- As you see in the recursive function **`insert`**, a new element will be inserted in a location, such that at the end in every sub-tree, the elements in the left child are smaller than the root, and the elements in the right child are greater than the root. Following is a sample binary tree:



Tasks:

- **Task 1:** Develop the **recursive** function **`inOrder`**, which **recursively traverses and prints** the elements of the binary tree in **increasing order**, i.e. **left child, parent, right child**. Print should be done into the output file. (See the sample outputs)
- **Task 2:** Develop the **recursive** function **`reverseOrder`**, which **recursively traverses and prints** the elements of the binary tree in **decreasing order**, i.e. **right child, parent, left child**. Print should be done into the output file. (See the sample outputs)
- **Task 3:** Develop the function **`levelOrder`**, which **traverses and prints** the elements of the tree **level by level**, from the top (root) to the bottom (leaves), left to right in each level. Print should be done into the output file. (**See the first algorithm on the next page and the sample outputs**.) **Hint: Use the two functions `enqueue` and `dequeue`.**
- **Task 4:** Develop the **recursive** function **`printTree`**, which **recursively traverses and prints** the elements of the tree, like a **tree with 90-degree counterclockwise rotated**. Print should be done into the output file. **(See the second algorithm on the next page and the sample outputs**.)
- **Task 5:** Develop the **recursive** function **`searchTree`**, which **recursively searches a value** in the binary tree. If the search is successful, it will return the corresponding node, NULL otherwise.
- **Task 6:** Inside the main function, based on the comments provided, **complete the code** by calling the proper functions and adding some other required statements.

To fulfill the above tasks, you must complete the C program, **`BinaryTree.c`**, that is provided.

* **Note**: You should first **CAREFULLY READ THE COMMENTS** provided for each part of the program, and then start completing it.

* **Note**: **DO NOT ALTER/DELETE** any part of the existing code. **ONLY complete the incomplete parts**.

An algorithm for **Level-Order traversal and print** the Binary Tree using a Queue

- **Enqueue** the root node into the queue
- While there are nodes left in the queue
    - **Dequeue** and **print** a value from the queue
    - If the pointer to the left child of the dequeued node is not **`NULL`**
        - **Enqueue** the left child node into the queue
    - If the pointer to the right child of the dequeued node is not **`NULL`**
        - **Enqueue** the right child node into the queue
- End While

A recursive algorithm for **printing a Binary Tree with 90-degree counterclockwise rotated**, using two arguments, tree root and number of preceding spaces to print, named **`totalSpaces`**, initialized with **zero**

- If the pointer to the current node is not **`NULL`**
    - Recursively call the function with the current node's **right subtree** and **`totalSpaces+5`**
    - Print spaces, based on the value of **`totalSpaces`**
    - Print the current node
    - Recursively call the function with the current node's **left subtree** and **`totalSpaces+5`**

Sample execution of the program:

Following is the content of the input file, `input.txt`:

```
12
6
15
21
2
8
18
6
3
9
27
```

Following is the content of the output file, `output.txt` after the program execution.

```
The numbers being inserted into the binary tree are:
12 6 15 21 2 8 18 6 duplicated 3 9 27
The Binary Tree in increasing order is:
2 3 6 8 9 12 15 18 21 27
The Binary Tree in reverse (decreasing) order is:
27 21 18 15 12 9 8 6 3 2
Level-Order traversal of the binary tree is:
12 6 15 2 8 21 3 9 18 27
Printing the binary tree with 90-degree counterclockwise rotated:

               27
          21
               18
     15
12
               9
          8
     6
               3
          2

Searching the binary tree:
6 found
25 not found
18 found
```

Question 2 (55 marks)

Ordered Doubly-LinkedList: In this question, you have been provided with an incomplete code for a Dynamic Ordered Doubly-LinkedList of strings. Your task is to complete the code by adding some functionalities to the program. Before describing the tasks, read the following initial notes for this question:

Initial Notes:

- This is an Ordered Doubly-LinkedList. This means
  - Every node has two links, one to the next node, **`nextPtr`**, and one to the previous node, **`prevPtr`**.
  - The previous link of the first node is **`NULL`**.
  - The next link of the last node is **`NULL`**.
  - It is an ordered list, which means the node's data values are increasing from the beginning to the end of the list.
- You must not only keep a pointer to the beginning of the LinkedList, **`startPtr`**, but also keep a pointer to the last node of the linkedlist, **`endPtr`**.
- The data part of every node is a **`char`** array of length **`15`** to store various names as strings.
- The data of node **`startPtr`** is set to **`"#"`**, and this node is not considered as a data node. The purpose of this node is that it points to the first data node of the LinkedList, and it always exists.
- The functions **`isEmpty`**, **`printList`**, and **`menu`**, have already been developed. **DO NOT ALTER them**. Just use them inside the other parts of the program that you must complete.

Tasks:

- **Task 1:** Develop the function **`setData`**, which sets the node's data, using a string value.
- **Task 2:** Develop the function **`insert`**, which inserts a new node into the doubly-linkedlist. Note that the link list should be in ascending order all the time. You should also take care of the starting and end pointers as well as the next and previous links for all the nodes affected.
- **Task 3:** Develop the function **`delete`**, which deletes the node with data equal to the given value. If it was successful, it will return the data, **`""`** otherwise. You should take care of the starting and end pointers as well as the next and previous links for all the nodes affected.
- **Task 4:** Develop the function **`printListReverse`**, which traverses and prints the elements of the linkedlist startging from the end, i.e. in descending order.
- **Task 5:** Inside the main function, based on the comments provided, complete the code by calling the proper functions and adding some other required statements.

To fulfill the above tasks, you must complete the C program, **`OrderedLinkedList.c`**, that is provided.

* **Note**: You should first **CAREFULLY READ THE COMMENTS** provided for each part of the program, and then start completing it.

* **Note**: **DO NOT ALTER/DELETE** any part of the existing code. **ONLY complete the incomplete parts**.

Sample execution of the program: **(Texts with Green color illustrate the user's inputs)**

```
An empty ordered Doubly-LinkedList created.
*****************************************
Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 to end.
? 1
Enter a name (Maximum 14 characters): Java
The list is:
NULL <--> Java <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 to end.
? 1
Enter a name (Maximum 14 characters): Python
The list is:
NULL <--> Java <--> Python <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 to end.
? 1
Enter a name (Maximum 14 characters): C++
The list is:
NULL <--> C++ <--> Java <--> Python <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 to end.
? 2
Enter character to be deleted: C#
C# not found.

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 to end.
? 2
```

```
Enter character to be deleted: C++
C++ deleted.
The list is:
NULL <--> Java <--> Python <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 to end.
? 4
The list in reverse order is:
NULL <--> Python <--> Java <--> NULL

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to print the list from the beginning.
   4 to print the list from the end.
   5 to end.
? 5
End of run.
```