Faculty of Arts & Sciences
Department of Computer Sciences
CMPS 212—Intermediate Programming with Data Structure
Quiz 2

Name: _____

Student ID: _____

- You have 100 minutes to complete this exam.
- This exam is closed-book (Except the 5 double sided A4 HANDWRITTEN Sheets). You may not use any computing devices including calculators.
- Code will be graded on proper behavior/output and not on style, unless otherwise indicated.
- If you enter the room, you must turn in an exam before leaving the room.
- You must show your Student ID to a TA or instructor for your exam to be accepted.
- **Some supporting material is provided at the end** (you can use the back pages for scratch)
- The problems are **NOT sorted by order of difficulty**

Good luck!

Score summary: (for grader only)

Problem	Description	Earned	Max
1	MC		24
2	Linked List Tracing		06
3	Heaps		12
4	Hashing		11
5	Trees		16
6	Recursive Programming		08
7	Linked List Programming		12
8	Binary Tree Programming		11
TOTAL	**Total Points**		**100**

Multiple Choices (24 points): Circle the most appropriate answer.

- What is big-O for the following pseudocode, given that `foo()` is `O(n)`, `bar()` is `O(log(n))`, and `condition()` is `O(n log(n))`?

```
for (int i = 0; i < n; ++i) {
      if (condition()) {
            foo();
      }
      else {
            bar();
      }
}
```

  - O(N)
  - O(logN)
  - O(NlogN)
  - O(N^2 logN)
  - O(N^3 log N)
  - O(N^2)
  - none of the above

- What is the overall runtime in Big-Oh notation of this algorithm?

```
int sum = 0;
for (int i = 1; i <= N*N; i++) {
    for (int j = 1; j <= i / 2; j += 2) {
        sum++;
    }
}
```

  - O(N)
  - O(N^2)
  - O(N^3)
  - O(NlogN)
  - O(N^2 logN)
  - none of the above

- An algorithm is O(n^2). It takes 20 sec for a given dataset. If twice as much data is used then
  - it will likely take 40 sec.
  - it will likely take 80 sec.
  - it will likely take 400 sec.
  - it will likely take 20 sec.
  - None of the above.

- The heapsort is _____ in the worst case.
  - O(*n*)
  - O(log *n*)
  - O(*n* log *n*)
  - O(n^2)
  - None of the above

- A stack is initially empty, then the following commands are performed:

  push 5, push 7, pop, push 10, push 5, pop

  which of the following is the correct stack after those commands (assume the top of the stack is on the left)?
  - 5 10 7 5
  - 5 10

- 7 5
- 10 5
- None of the above

- To remove a node N from a linear linked list, you will need to _____.
  - set the pointer `next` in the node that precedes N to point to the node that follows N
  - set the pointer `next` in the node that precedes N to point to N
  - set the pointer `next` in the node that follows N to point to the node that precedes N
  - set the pointer `next` in N to point to the node that follows N
  - None of the above

- Which data structure represents a waiting line and limits insertions to be made at the back of the data structure and limits removals to be made from the front?
  - Stack.
  - tree.
  - Linked list
  - Queue.
  - None of the above

- Which of the following is *not* a similarity between a stack and a queue as described by the text?
  - both have an `isEmpty` method
  - both can retrieve an item from any position
  - both can insert an entry at one end of the structure
  - both can retrieve an entry from one end of the structure
  - None of the above

- The traversal of a binary tree (having n nodes) is _____.
  - O(n)
  - O(1)
  - O(n^2)
  - O($\log_2 n$)
  - None of the above

- The maximum height of a binary tree of n nodes is _____.
  - n
  - $n / 2$
  - $(n / 2) - 2$
  - $\log_2(n + 1)$
  - None of the above

- In an array-based implementation of a heap, the `remove` operation is _____.
  - O(1)
  - O(n)
  - O(n^2)
  - O($\log n$)
  - None of the above

- Which of the following is always true when adding an element to a heap?
  - The new element will always be a leaf.
  - The new element will always be the root.
  - The new element will always be an internal node.
  - The new element will always have 2 children.
  - none of the above is always true

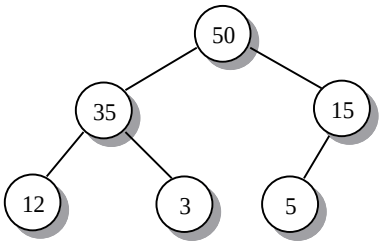
Linked List Tracing (06 pts)

What is the output produced by this code?

```
class ListNode {
    public:
            int data;
            ListNode *next;
          // other methods
}


class LinkedIntList {
   private:
       ListNode *head;

   public:
   void foo() {
      if (head != NULL) {
            ListNode *current = head;
            while (current->next != NULL) {
                  if (current->next->data < 0) {
                              ListNode temp = current->next;
                              current->next = current->next->next;
                              temp->next = head;
                              head = temp;
                  } else {
                              current = current-->next;
                  }
            }
      }
   }
};


int main () {
      // omitted code:
      //build a linkedList to contain the following nodes in the following:
      // list = [8, 7, -4, 19, 0, 43, -8, -7, 2]
      // 8→7→-4→19→0→43→-8→-7→2

      list.foo();
      //show the order of the nodes after executing list.foo();
}
```

Heaps (12 pts)

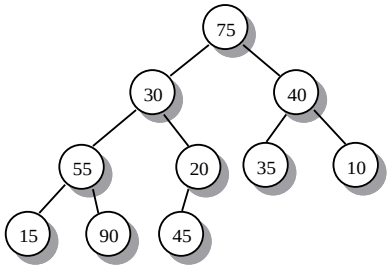
A Max heap can be represented by a vector. Start with the following heap and list the elements after each operation. Execute the operations sequentially, using the result of the previous operation. The initial vector values for the heap are {50, 35, 15, 12, 3, 5}.



(a) Insert 23: The vector values are now {_____}.

(b) Erase an element from the heap: The vector values are now {_____}.

Start with following tree and **"heapify"** it to create a minimum heap. Draw the resulting tree.



Hashing (11 pts)

The following hash table is used to store integer values.
Assume that we use the following hashing function to store values in the table.

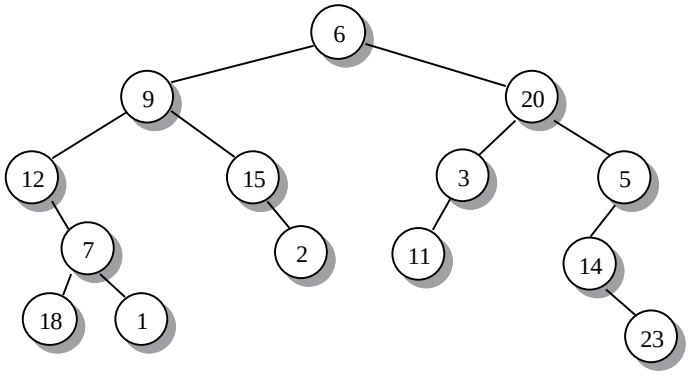
`h(x) = x % someInt` (where `someInt` is equal to the size of the hash table below - which is **10**)

Show the resultant hash table after inserting the values: 12, 11, 2, 7, 10, 21, 121, 23, 33, 71, 90 in that order.
Use the separate chaining technique (with linked lists) for collision resolution.

Index	Value
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Trees (16 pts)

- For each subpart (a) – (g), use the following tree.



(a) (1 point) What is the parent of node 5? _____

(b) (2 points) What is the depth of this tree? _____

(c) (1 point) List the nodes in subtree 12. _____

(d) (2 points) List all of the non-leaf nodes. _____

(e) (2 points) Give the order that the nodes are visited in a preorder traversal of the tree.
_____

(f) (2 points) Give the order that the nodes are visited in an inorder traversal of the tree.
_____

(g) (2 points) Give the order that the nodes are visited in an postorder traversal of the tree.
_____

- (4 points) The post-order traversal of a binary search tree gives: 20 35 30 41 45 42 40 53 58 60 55 52 .
  Draw a diagram of the tree.

Recursive Programming (8 points)

Write a recursive function called parenthesize that takes a string and an integer n as parameters and that prints the string inside n sets of parentheses. For example, this code:

```
parenthesize("Joe", 2);
cout<<endl;
parenthesize("AUB", 6);
cout<<endl;
parenthesize("QUIZ 2", 1);
cout<<endl;
```

should produce these 3 lines of output:

```
((Joe))
((((((AUB))))))
(QUIZ 2)
```

Your method should throw or output "Error: bad parameter value" if passed a negative number. It could be passed 0, as in:

```
parenthesize("CMPS 212, Spring 2018-19", 0);
```

In this case the output would have no (i.e., 0) parentheses:

```
CMPS 212, Spring 2018-19
```

You are not allowed to use any variable except the function parameters.

Linked list programming (12 points)

Write a function that takes a pointer to the front of a linked list of doubles, as well as a target value and a threshold. It should remove all values in the list that are within the threshold of the target value. For example, suppose you are given the following list, a target of 3.0, and a threshold of 0.3:

{3.0, 9.0, 4.2, 2.1, 3.3, 2.3, 3.4, 4.0, 2.9, 2.7, 3.1, 18.2}

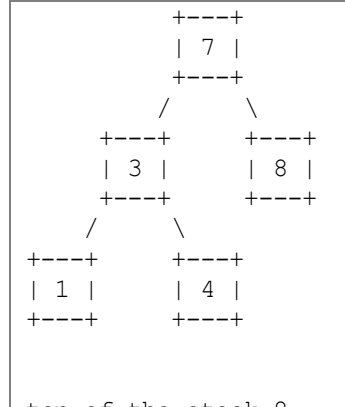
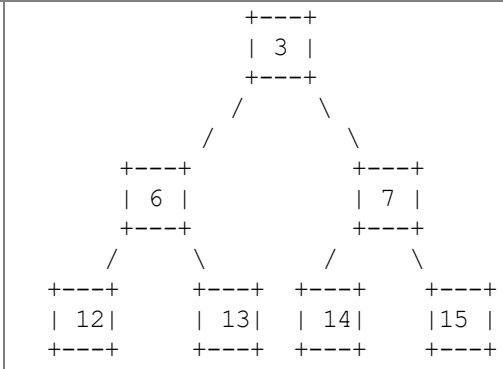
You should remove the underlined values, yielding the following list:

{9.0, 4.2, 2.1, 2.3, 3.4, 4.0, 18.2}

If the list is empty or values within the given range don't appear in the list, then the list should not be changed by your function. You should preserve the original order of the list.

Binary Tree Programming (11 points)

Write a **recursive** method **`treeToStack`** that could be added to the **`LinkedBinaryTree`** class. **`treeToStack`** method should push all the data values of a tree into a **stack**. If the tree is a **binary search tree**, then the lowest data value should be at the bottom of the stack and the highest value should be at the top (in other words, the values should appear in the stack from bottom to top in increasing order). **`treeToStack`** should return a **`Stack`** of integers.

BST tree	Binary tree
 top of the stack=8 Bottom of the stack=1	 top of the stack=15 Bottom of the stack=12

You may define private helper methods to solve this problem, but otherwise you may not create any data structures such as arrays, lists, etc. Your method should not change the structure or contents of either of the two original trees being examined.

Supporting Material

Stack

- **`size()`:** Return the number of elements in the stack.
- **`empty()`:** Return true if the stack is empty and false otherwise.
- **`push(e)`:** Push *e* onto the top of the stack.
- **`pop()`:** Pop the element at the top of the stack.
- **`top()`:** Return a reference to the element at the top of the stack.

Binary Tree

```
struct Node {     // a node of the tree
    int elt;      // element value
    Node* left;   // left child
    Node* right;  // right child
};

class LinkedBinaryTree {
public:
    …
private:
    Node* root;               // pointer to the root
    int n;              // number of nodes
};
```

Linked List

```
struct DoubleNode {
   double data;
   DoubleNode *next;
}

class LinkedList {
    private:
       DoubleNode *head;
    public:
   …
   void removeAllThreshold(DoubleNode *&front, double value, double threshold);
};
```