CPSC 221 2017W1: Midterm Exam

October 18, 2017

1 Who gets the marks? [1 marks]

Please enter your 4 or 5 digit CSID in this box:

2 Choices, Choices! [12 marks]

MC1 [1.5 marks]

Consider this simple example, and assume the standard `iostream` library has been included.

```
1  void doub(int x) { x = x * 2; }
2  void trip(int * x) { *x = *x * 3; }
3  void quin(int & x) { x = x * 5; }
4
5  int main() {
6      int y = 7;
7
8      doub(y);
9      trip(&y);
10     quin(y);
11
12     cout << y << endl;
13     return 0;
14 }
```

What is output on line 12?

- ◯ 7
- ◯ 35
- ◯ 70
- ◯ 105
- ◯ 210
- ◯ This code does not compile.

MC2 [1.5 marks]

In the example below, assume the `sphere` class has public member functions `setRadius` and `getDiameter`, and a one argument constructor. Also assume that the `iostream` library has been included.

```
sphere *a, *b;

a = new sphere(1.0);
b = a;
b->setRadius(2.0);
delete b;

a->setRadius(4.0);
sphere * c = new sphere(5.0);
b = new sphere(3.0);

cout << a->getDiameter() << endl;
```

What is the result of executing these statements?

- ◯ 6.0 is sent to standard out.
- ◯ 8.0 is sent to standard out.
- ◯ 10.0 is sent to standard out.
- ◯ This code exhibits unpredictable behavior.
- ◯ This code does not compile.
- ◯ This code has a memory leak.

MC3 [1.5 marks]

Consider a class `List` that is implemented using a singly linked list with a `head` pointer aimed at the first element of the list, and assembled in such a way that the `next` pointer of the last node has the same value as `head`. This structure is commonly referred to as a *circular linked list*. Given that representation, which of the following operations could be implemented in $O(1)$ time? Select all that apply. (Assume the list is sufficiently long for all the options to be valid.)

- ◯ Insert an item at the front of the list.
- ◯ Insert an item at the third position of the list.
- ◯ Insert an item at the middle position of the list.
- ◯ Insert an item at the end of the list.
- ◯ Delete the front item from list.
- ◯ Delete the third item from list.
- ◯ Delete the last item from list.

MC4 [1.5 marks]

Fill in the blanks so that the following sentence is true: If you have a complete binary tree with 333 nodes, the height (h) of the tree is ________ and there are ________ nodes on level h.

- ◯ First blank is 7, second is 78.
- ◯ First blank is 7, second is 79.
- ◯ First blank is 8, second is 78.
- ◯ First blank is 8, second is 79.
- ◯ First blank is 9, second is 78.
- ◯ First blank is 9, second is 79.
- ◯ None of the other options makes the sentence true.

MC5 [1.5 marks]

Consider the binary tree class with 1) variable `root` that is the `Node` representing the root of the binary tree and 2) each `Node` consists of an integer `data` element, and two `Node` pointers called `left` and `right`.

What does `fun(root)` return?

```
int fun(Node * curr) {
  if (curr != null) {
    int ret1 = fun(curr->left);
    int ret2 = fun(curr->right);
    return 1 + ret1 + ret2;
  }
  else return 0;
}
```

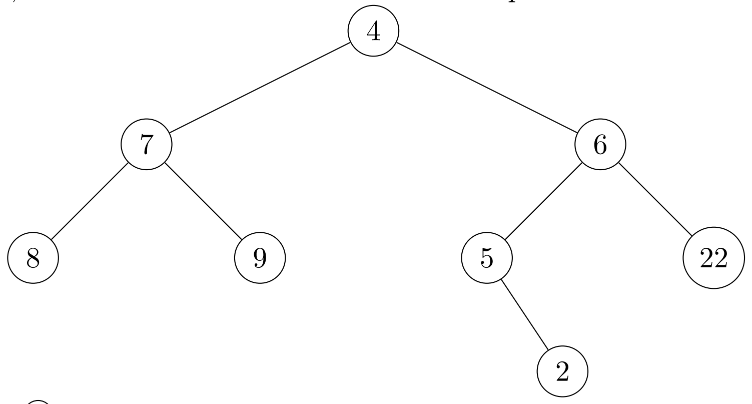
◯ `fun` returns the height of the tree.
◯ `fun` returns the number of keys in the tree.
◯ `fun` returns the sum of all keys in the tree.
◯ `fun` returns the shortest distance from root to leaf.
◯ None of the other options is correct.

MC6 [1.5 marks]

Suppose that you have a binary tree of height h containing n nodes. What is the running time of an efficient algorithm to determine if the tree is in order. (i.e at every node k, all the elements in the k's left subtree are smaller or equal to k's key and all the elements in the k's right subtree are larger or equal to k's key.)
◯ $O(1)$
◯ $O(h)$
◯ $O(n)$
◯ $O(n^2)$
◯ None of the options is correct.

MC7 [1.5 marks]

Suppose you implement a variation of a level order traversal in which the children of a node are enqueued from right to left, rather than the usual left to right. In this new level order traversal of the following binary tree, which is the last node that will be dequeued before the node 9 is enqueued?



◯ 4
◯ 7
◯ 6
◯ 8
◯ None of the options is correct

MC8 [1.5 marks]

What is the upper bound on the running time of an algorithm to perform a level-order traversal of a binary tree containing n nodes?
◯ $O(1)$
◯ $O(\log n)$
◯ $O(n)$
◯ $O(n \log n)$
◯ $O(n^2)$

3 SlideSort [12 marks]

We have discussed 3 different iterative sorting algorithms: selection, insertion, and shell. Here is code for a fourth, which we're calling "Slide Sort". (The `swap` function simply switches the values of the two parameters.)

```
void slideSort(vector<int> & a){
   int n = a.size();
   for (int i = n-1; i >=0 ; i--){
      for (int j = 0 ; j < i; j++){
         if (a[j] > a[j+1])
            swap(a[j],a[j+1]);
      }
   }
}
```

1. **[4 marks]** Use the table below to trace the first 2 iterations on i, given the initial values in the array.

	0	1	2	3	4	5	6	7
start	4	3	7	6	0	5	2	1
after i=7								
after i=6								

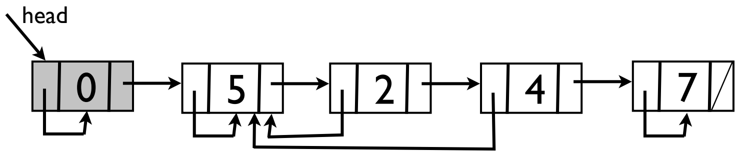
2. **[4 marks]** Which of these invariants does the above code satisfy at the end of each outer `i` loop? Choose all that apply.

   ◯ Entries `a[0]` through `a[i]` are in ascending order.
   ◯ Entries `a[0]` through `a[i]` contain the smallest keys in the entire array.
   ◯ Entries `a[i]` through `a[n-1]` are in ascending order.
   ◯ Entries `a[i]` through `a[n-1]` contain the largest keys in the entire array.

3. **[4 marks]** Give an expression for the number of times the comparison in line 5 is executed when the code is deployed on an array containing n integers. Be as accurate as you can (do not use asymptotic notation).

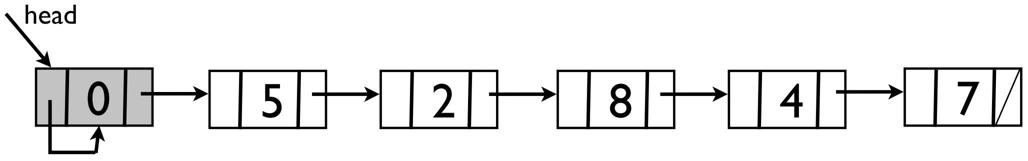
4 MaxList [17 marks]

In this problem we will investigate a variation of a linked list that we will call a `MaxList`. As illustrated in the figure below, this list has a `head` pointer and a *zero node* containing the value 0. In addition, each `MaxListNode` has non-negative integer `data`, a `next` pointer, and a `maxPrev` pointer that points to the maximum valued node before and including the current node. Look carefully at the picture to make sure you understand the structure. You may assume that all of the data elements are unique.

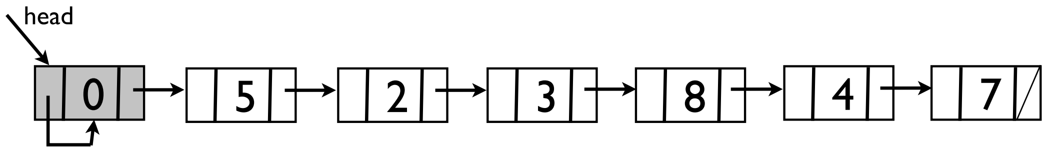


A partial definition of the `MaxList` class can be found at the end of this exam. Please tear it off and use it to help you answer the following questions.

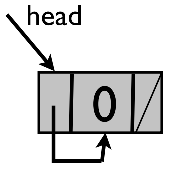
1. **[2 marks]** In this diagram, we have just invoked the function call `insertAt(3,8)` which places the value 8 in the third position in the list. Draw the accurate `maxPrev` pointers, notice which ones may change (depending on the value inserted), and which ones will certainly not change.



2. **[2 marks]** In this diagram, we have just invoked the function call `insertAt(3,3)` which now places the value 3 in the third position in the list. Draw the accurate `maxPrev` pointers. In this example, you should pay particularly close attention to how the `maxPrev` pointer is set for the new node.



3. **[3 marks]** The diagram below illustrates the state of memory after an empty `MaxList` is declared (i.e. `MaxList fancyList;`). Write the no-argument constructor for the `MaxList` class as it would appear in the maxlist.cpp file. Note that the `maxPrev` pointer is set to point to the zero node itself, which means that the `maxPrev` pointer is set correctly for that node (whose data value is 0). The length of an empty `MaxList` is 0.



```
MaxList::MaxList() {
```

4. **[5 marks]** Implement the public `insertAt` function so that it behaves as described in parts (a) and (b) of this problem. For simplicity, you may assume that the list is always long enough to accept a new value in position `k`. As illustrated above, the first data (non-sentinel) element occupies position 1, the second occupies position 2, etc. You should use some of the private member functions described in the `MaxList` class definition. If you feel like you need more space than we've given you, then you may want to rethink your approach to the problem.

```
void MaxList::insertAt(int k, int ndata){






}
```

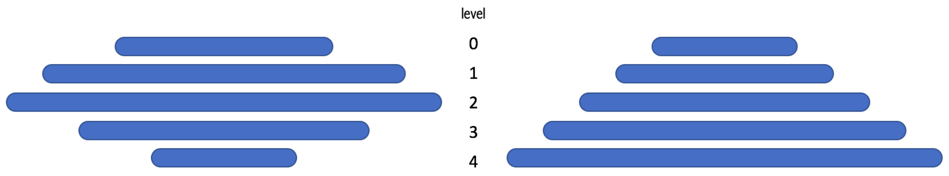
5. **[2 marks]** Imagine an implementation of a function called `insertAtFront` that adds a new value to the front of the list. What is the running time of such a function in the worst case? (Circle the appropriate answer.)

$$O(1) \quad O(\log n) \quad O(n) \quad O(n \log n)$$

6. **[3 marks]** Briefly explain why the `find` function is in the `private:` section of the class definition.

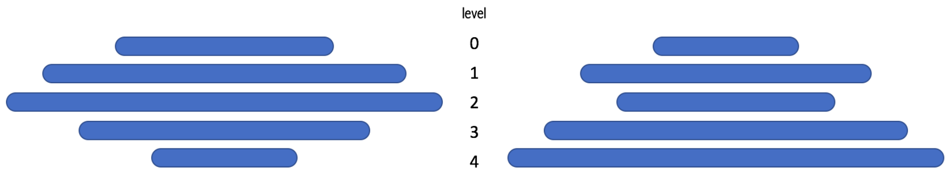
5 Pancakes [16 marks]

Suppose we have a messy stack of pancakes like those on the left, and we'd like to arrange them neatly, like those on the right. Unfortunately, we can't just sort them, because pancakes can only be maneuvered using a spatula, and they must stay in a stack. Our goal is to sort the pancakes using only an operation called a flip, which corresponds to inserting a spatula beneath some pancake, lifting that pancake and all those above it, inverting the entire group, and replacing the inverted group back onto the original stack.



Every pancake is represented by a real number that corresponds to its radius, and a stack of n pancakes is represented by a `stack` of those radii, with the top pancake at the top of the stack. The `flip` function, when parameterized by level k, reverses the elements of the stack between the top and the k^{th}, inclusive.

1. **[2 marks]** Describe a sequence of 2 flips that will transform the stack on the left to the stack on the right:



```
flip(____);
flip(____);
```

2. **[4 marks]** Complete the function `flapjack`. It takes a stack of pancake radii, and rearranges the entries so that they are sorted in ascending order. You may assume the following two functions exist: `flip(stack<double> & A, int k)` - reverses the entries as described above. and `findMax(stack<double> & A, int a, int b)` - returns the position (level) of the largest pancake in the stack between indices `a` and `b`, inclusive.

```
void flapjack(stack<double> & A) {

  for (int i = ________; i ________; i________){

    int k = findMax(________, ________, ________);

    flip(________, ________);

    flip(________, ________);
  }
}
```

3. **[6 marks]** Complete the `flip` function. Your implementation is constrained to use only one local data structure–a stack or a queue, and the iteration variable of a `for` loop. You may assume that the `Stack` interface includes public `push`, `pop`, and `peek` functions, and that the `Queue` interface includes public `enqueue`, `dequeue`, and `peek` functions.

```
void flip(stack<double> & A, int k) {










}
```

4. **[4 marks]** *Circle* a least upper bound on the number of calls to `flip` performed by `flapjack`, and *underline* a least upper bound on the total running time of `flapjack` assuming `findMax` runs in $O(n)$ time.

$$O(1) \quad O(\log n) \quad O(n) \quad O(n \log n) \quad O(n^2) \quad O(n^3)$$

6 Shade [14 marks]

1. On his way to class last Tuesday, Geoff dropped his binary tree, shattering it into a bazillion pieces. Lucky for him, he had written the pre-order and in-order traversals of the tree in his notes. We will use this information to rebuild the tree.

preOrder:	D	R	A	G	O	N	S

inOrder:	R	A	D	N	O	G	S

   (a) **[1 marks]** Which key corresponds to the root of the tree? Circle it on both tables.

   (b) **[1 marks]** Which keys must be in the left subtree? Underline those on both tables.

   (c) **[1 marks]** Which keys are in the right subtree? Box those on both tables.

   (d) **[3 marks]** Finally, use the fact that binary trees are recursively defined structures to finish building the tree.

2. Unfortunately, the same thing happened again today! This time Geoff only had a record of the post-order traversal of his tree, but he remembered that it was a binary *search* tree (ordered alphabetically). Use the post-order traversal given below, together with the fact that the broken tree is a BST, to reconstruct the tree.

postOrder:	H	G	N	K	S	T	I

   (a) **[2 marks]** The fact that the tree is a BST allows you to infer something about one of the traversals. Write the name of the traversal, and everything you know about it in the table below.

___Order:							

   (b) **[1 marks]** Which key corresponds to the root of the tree? Circle it on both tables.

   (c) **[1 marks]** Which keys must be in the left subtree? Underline those on both tables.

   (d) **[1 marks]** Which keys are in the right subtree? Box those on both tables.

   (e) **[3 marks]** Finally, use the fact that binary trees are recursively defined structures to finish building the tree.

This page intentionally left (almost) blank.

If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.

This page intentionally left (almost) blank.
If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.

The following code is a partial definition of the `MaxList` class.

```
class MaxList  {
public:
    MaxList();  // YOU'LL DEFINE

    // the big 3: (given) copy ctor, dtor, and op=
    int  size() const; // (given) the number of data elts in list

    // insertAt:   inserts data at position k of list, updating maxPrev
    //             pointers appropriately.  Assumes k is a valid position.
    void insertAt(int k, int ndata);   // YOU'LL DEFINE

private:
    class MaxListNode  {
    public:
        MaxListNode(int ndata);
        MaxListNode * next;
        MaxListNode * maxPrev;
        int data;
    };

    MaxListNode * head;
    int length;

    //find: (given) walks k steps from curr
    MaxListNode * find(int k, MaxListNode * curr);

    // fixPtrsAfter: (given)  takes a pointer to a MaxListNode whose
    //             maxPrev pointer is set correctly, and updates all maxPrev
    //             pointers from there through the end of the list.
    void fixPtrsAfter(MaxListNode * curr);

    // insertAfter: (given)  creates a new node with value ndata, and inserts
    //             it into the list after the node pointed to by parameter
    //             curr.  It does not update any maxPrev pointers.
    void insertAfter(MaxListNode * curr, int ndata);
};
```