CSCI 2270 Exam 2
Spring 2013

Exam 1 topics plus B-Trees, Priority Queues, Huffman Encoding

Write Clearly!

Name: _____

9-Digit Student ID: _____

TA (circle one):
Xin Yogesh Shang Don't Know

1. Recursion (5 points): Two parts to this question. First: What is the output when we call thing(5)? Second: What is the output when we call otherthing(5)?

```
void thing(int ttl) {
  if (ttl > 0) {
    cout << ttl << endl;
    thing(ttl - 1);
  }
}

void otherthing(int ttl) {
  if (ttl > 0) {
    otherthing(ttl - 1);
    cout << ttl << endl;
  }
}
```

thing output

5
4
3
2
1

otherthing output

1
2
3
4
5

2. Pointers (5 points): Say we have a struct called **goo**, defined on the right.

```
struct goo {
  goo* a;
  goo** b;
};
```

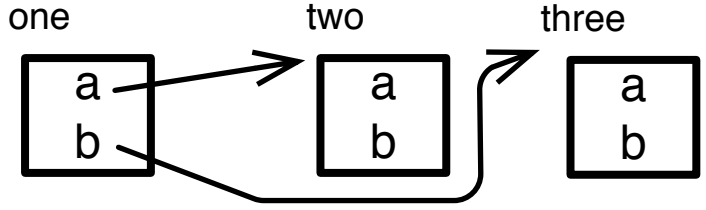
(a) (2 points) What's the data type of x?

goo* x = new goo;

(b) (3 points) Let's say we have three variables of type **goo*** called **one**, **two**, and **three**. Write the C++ code that makes **one**'s **a** member point to **two**, and **one**'s **b** member point to **three**, as in the diagram at the bottom.

```
one->a = two;
one->b = &three;
```

remember, & is the address-of operator.



_____ / 10

3. Pointers (6 points): What does the code at right print out?

15 5
15 71

```
void sputnik(int& a, int b) {
  a = a * b;
  b = a - 4;
}

void kili(int a, int& b) {
  a = a * b;
  b = a - 4;
}

int main() {
  int a = 3;
  int b = 5;
  sputnik(a, b);
  cout << a << " " << b << endl;
  kili(a, b);
  cout << a << " " << b << endl;
}
```

4. Complexity (6 points): What is the average runtime complexity of the following operations? (2 pts each)

(a) access first element of a heap?

O(1)

(b) insert one item into a binary tree?

O(log n), or $O(\log_2 n)$

(c) insert one item into a B-Tree of order **m**?

$O(\log_m n)$

5. Priority Queue (6 points): Say we create an empty priority queue **Q** that prioritizes large values over small ones. We add and remove things from it as follows. **pop()** removes and returns the highest priority value, **peek()** returns (but does not remove) it, and **add(v)** adds the value **v** at the appropriate location. Write the contents of the queue in their priority order after this sequence of operations.

```
Q = (empty priority queue)
Q.add(8)
Q.add(10)
Q.add(4)
Q.pop()
Q.peek()
Q.add(15)
Q.add(14)
Q.pop()
Q.pop()
```

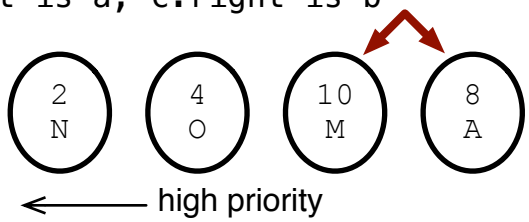
8, 4

if you wrote 4, 8 but indicated the 8 was the high priority end, that is ok too.

6. PQ & Huffman (22 points): *This is a cumulative question. You have to get the previous one right in order to get points for later questions.* Huffman encoding uses a priority queue to build an encoding tree based on frequency information per symbol, where low frequencies have higher priority. Supernodes combine lower-level nodes, and its frequency is the sum of the children. The algorithm could look like this:

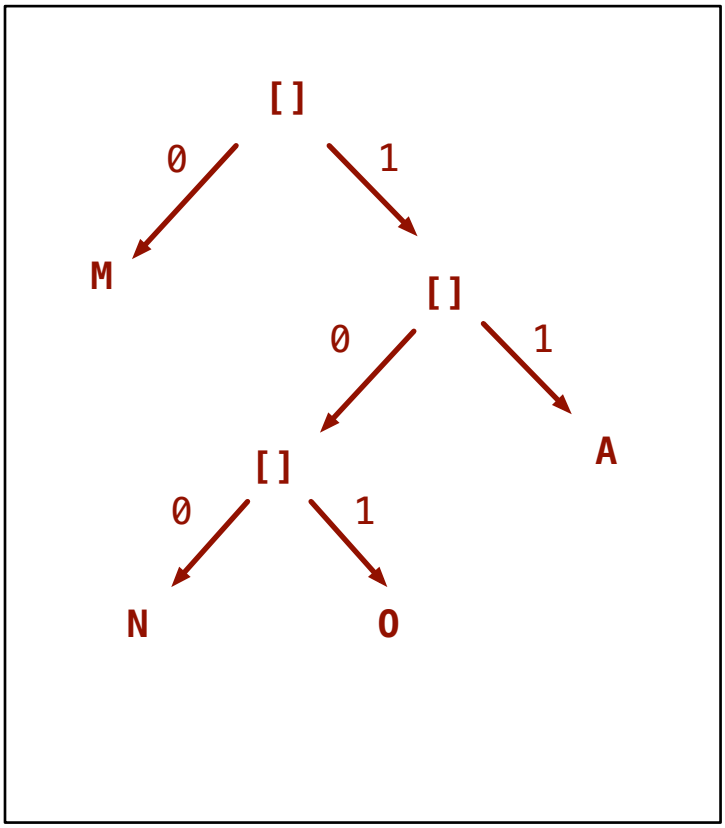
```
pq = <priority queue with 100 leaf nodes>
while (pq has more than one element):
    a = pq.pop()   -- this removes highest priority item & returns it
    b = pq.pop()
    c = build_supernode() -- here, c.left is a, c.right is b
    pq.insert(c)
    say_hello()
huffman_root = pq.pop()
```

Say our initial priority queue looks like the diagram at the right.



(a) (4 points) There is an error in the ordering of nodes in this priority queue. You can fix the bug by swapping two elements. Indicate which two should swap. **Swap the 10/M and 8/A nodes.**

(b) (6 points) Execute the above tree-building algorithm using your corrected ordering of the priority queue. Read the comment next to **build_supernode()**, because that is what guarantees that the answer is unique. Draw the resulting encoding tree in the box below.



Key for c & d

0 = left
1 = right

(c) (6 pts) Fill the encoding map:

char	bitstring
O	**101**
A	**11**
M	**0**
N	**100**

(d) (6 pts) Decode the following bitstring back into English:

Encoded: 0101101100011100

Decoded: **MOONMAN**

_____ / 22