ECE 250
Data Structures and Algorithms
QUIZ 4

2006-11-20

The quiz is out of 21 marks.
No questions, no aides.
If you are unsure about a question, write down your assumptions and continue.
This examination has two pages of questions.
If you run out of room, use the reverse of this page.

Surname, Given Name	**Student ID**

Sign here to indicate that you have read the above instructions.

1. **[3]** Insert 003, 513, 424, 637, 246, 155 into the following into the following initially-empty hash table by using a hash function which maps an integer to its least-significant digit. Use linear probing if a collision occurs.

0	1	2	3	4	5	6	7	8	9

2. **[3]** Insert 003, 513, 424, 637, 246, 155 into the following into the following initially-empty hash table by using a hash function which maps an integer to its least-significant digit. Use double hashing if a collision occurs where the second hash function is the appropriately-modified (as discussed in class) second-least-significant digit.

0	1	2	3	4	5	6	7

3. [2] In a min heap with *n* elements, what is the number of nodes which must be searched to find the maximum element of the heap? Write your answer as if you were using C++-style integer division (which truncates any fractional part).

4. [3] Given the min heap

	2	3	5	9	8	12	4

find the state of the min heap after one dequeue of the minimum element using the algorithm shown in class. Enter your answer into the following table:

5. [2] Given the unsorted array

5	3	6	2	8	1	4	7	9	0

perform one pass through the array as defined by the bubble sort algorithm. Enter your answer in this table:

									9

6. [4] Convert the following unsorted list into a max-heap using the **O**(*n*) heapification process shown in class:

7	8	4	2	3	1	5	6	9	0

Enter your answer in this table:

7. [4] In Project 3, you implemented an expression tree for integers. The expression class had a member variable root which is a pointer to an instance of the expression tree class. The constructor creates an instance of the expression tree class storing the integer 0.

```
class Expression {
        private:
                ExpressionTree * root;

        public:
                Expression( int = 0) );
                ~Expression();

                void add( int );        // this + n
                void subtract( int );   // this - n
};
```

As with Project 3, implement the functions add and subtract but with the following changes:

1. If you are adding a negative number to **`this`**, convert it to a subtraction of the absolute value of the number.
2. If you are subtracting a negative number from this, convert it into a sum of **`this`** and the absolute value of the number.
3. Leave the expression tree unchanged if you are adding or subtracting 0.

The relevant elements of the expression tree class are:

```
class ExpressionTree {
  public:
    ExpressionTree( int = 0, ExpressionTree * = 0, ExpressionTree * = 0
  );
    ~ExpressionTree();

    static const int PLUS = 0;    // You must use these
    static const int MINUS = 1;
    static const int TIMES = 2;
    static const int DIVIDE = 3;
};
```

```
void Expression::add( int n ) {

}
```

```
void ExpressionTree::subtract( int n ) {

}
```