

COP 3530
Data Structures

Midsemester Exam

Name: _____

Email: _____

October 13, 1997

This exam has 4 questions. Each question starts on a new page. Each page is worth 50 points. Please answer each question on its page. You may write on the back of a page.

1. This question refers to the function `FindMatch` defined below. `FindMatch` returns true if there is any integer in `A` that is also in `B`. Assume that both arrays have N elements. Ignore all syntax errors.

```
int FindMatch( const Vector<int> & A, const Vector<int> & B )
{
    for( int i = 0; i < A.Length( ); i++ )
        for( int j = 0; j < B.Length( ); j++ )
            if( A[ i ] == B[ j ] )
                return 1;

    return 0;
}
```

- (a) What is the running time of this function?
- (b) If it takes ten seconds to run `FindMatch` on an array of 100 elements, approximately how long will it take to run on an array of 400 elements?
- (c) Describe a more efficient algorithm than the one above and give its running time. You may not use any code in your description.

2. Write a complete generic stack class using an efficient implementation of your choice. If you use an `Vector` implementation, you must include `Vector-doubling` code.

3. An `Object` is a complex and unknown entity, for which you may assume the existence of the following: a copy constructor, `operator=`, and a function called

```
int IsGood( ) const;    // Returns true if good, false otherwise.
```

Describe an algorithm that accepts as parameter a `Vector<Object>` and rearranges the `Vector` so that all `Objects` that are *good* (as determined by `IsGood`) precede all objects that are *false*. After describing the algorithm in English, provide working C++ code.

Your algorithm must satisfy the following requirements:

- (a) It must run in linear worst-case time.
- (b) It may not use more than constant extra memory. In other words, you may not use a second vector.
- (c) It may only use members of `Object` that are specifically mentioned above.

4. (a) N items are inserted into an initially empty binary search tree. What is the time spent **for the last operation**
 - i. on average
 - ii. in the worst case
- (b) Among insertion sort, shellsort, mergesort, and quicksort which algorithm(s):
 - i. Has the best worst-case performance (as measured by Big-Oh)?
 - ii. Has the poorest average-case performance?
 - iii. Uses constant extra space?
 - iv. Does not naturally use recursion?
 - v. Performs the best when presented with already-sorted data?