

CS 301: Midterm Exam #1

Problem 0 (5 points): Write your name on the front of your answer booklet.

Problem 1 (25 points): Write a C++ class called `Pair` with the following members. The data members (instance variables) should be private; everything else should be public. Write both the header file and the implementation file. Declare methods and parameters as `const` as appropriate.

- The class should have a single data member that is a statically allocated array of two `ints`;
- it should have a constructor that initializes both elements of the array to zero;
- it should have a method `copyFrom` that takes a `Pair` as a parameter and copies the two elements from that object's array into the array inside the object the method was invoked on; and
- it should have a method that takes an output stream as a parameter (either standard output or an output file) and writes the two elements of the array to that stream.

Problem 2 (25 points):

- (a) Consider an implementation of a sorted list ADT that allows duplicate keys. Modify the given `binarySearch` so it returns either the index of the first element whose key is greater than or equal to the given key, or returns the length of the list if there is no such element. See the last page for the prototype. `binarySearchGE` should still execute in $O(\log n)$ time. (Hint: take care of special cases first and then let your loop invariant be that the desired element is between array indices `first` and `last` inclusive.)
- (b) Assuming you have a working version of `binarySearchGE` and a corresponding method called `binarySearchGT` that searches for the index of the first element whose key is strictly greater than the given key, write a member function for the given sorted list implementation called `count` that counts how many items in the list have a key matching the given key. Full credit will be given only for $O(\log n)$ algorithms.

Problem 3 (20 points): Suppose the following blocks of code both do the same thing.

```
// Block A
X x;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        x.foo(n);

// Block B
X x;
for (int i = 0; i < n; i++)
    x.bar(n);
```

Suppose implementation 1 of class **X** has a **foo** method that runs in $O(n)$ time and a **bar** method that runs in $O(n \log n)$ time. Suppose implementation 2 of class **X** has a **foo** method that runs in $O(1)$ time and a **bar** method that runs in $O(n^2)$ time. In terms of asymptotic efficiency, which combination of algorithm (block A or B) and implementation (implementation 1 or 2) is best? Explain your answer.

Problem 4 (25 points): Consider the implementation of the queue ADT discussed in class.

- (a) Add a method **reverse** to that implementation. **reverse** should reverse the order of the items on the queue; it takes no parameters and returns **void**. For full credit, do not use a significant amount of temporary storage space (in mathematical terms, the total size of your local variables should be $O(1)$). If you do use temporary storage space such as a stack or array, you will receive partial credit.
- (b) Compute the asymptotic running time of your **reverse** method.