CSE250, Spring 2014 **Prelim I** **Name:**________________________
Mar. 10, 2014 **St.ID#:**________________________

Closed book, closed-notes-except-for-1-sheet, closed neighbors, 48 minutes. This question paper has FOUR pages. Do ALL THREE questions **on this exam paper**. Please *show all your work*; this may help for partial credit. The exam totals 67 pts., subdivided as shown.

(1) (3+6+1+9 = 19 pts.)
Consider the following declarations in C++:

```
string a = "They call me ";
const string* apcd = new string("Yellow ");
string* const acp = new string(*apcd);
string* ap = acp;
```

(i) Which of the following two statements is legal? Write the legal one.

```
apcd->at(0) = 'M';
acp->at(0) = 'M';
```

(ii) After the legal change, say what the following two lines print:

```
cout << a << *acp << *ap << endl;
cout << a << *acp << *apcd << endl;
```

(iii) In what legal statement above is a *copy* of a string being made? Write the statement out.

(iv) For each of the following statements, say whether it would compile, and if not, explain what the error is (you need not remember what g++ would say).

(a) `a.at(0) = 'W';`

(b) `ap->at(0) = 'H';`

(c) `apcd = new string("Bellow");`

(d) `acp = &a;`

(e) `ap = a;`

Bonus: For 2 pts. exam extra-credit, explain the difference when you execute `ap->at(-1) = 'Q';` compared to running `(*ap)[-1] = 'Q';`

(2) (6+6+3+9 = 24 pts.)

The sketches of code labeled (a)–(d) on the next page all involve a function `int sum(vector<int> v, int k, int m)` which returns $\sum_{i=k}^{m} v(i)$. Note that with the standard assumption that `int` arithmetic operations take $O(1)$ time, `sum` runs in $O(m-k)$ time—or in precise notation, in $\Theta(m-k)$ time. Here is the `sum` function in full, with a const-alias on the vector parameter:

```
int sum(const vector<int>& v, int k, int m) {
    int total = 0;
    for (int i = k; i < m; i++) {
        total += v.at(i);
    }
    return total;
}
```

(i) For (a) and (b) below, state the exact number of times the body of the for-loop is executed. (You may assume n is even.)

(ii) Give asymptotic running times for (a) and (b), using precise asymptotic notation.

(iii) And give an asymptotic running time for the recursive function `foo4` in (d).

(iv) Now rank (a)–(d) in order from asymptotically fastest to slowest, i.e. in order of little-o notation. (You do not need to know or give the particular asymptotic running time of (c) to answer this.)

```
(a) int bar1(vector<int> v, int n) {
       int acc = 0;
       for (int m = 0; m < n; m++) {
          acc += sum(v,0,m);
       }
       return acc;
    }

(b) int bar2(vector<int> v, int n) {
       int acc = 0;
       for (int i = 0; i < n/2; i++) {
          for (int j = n/2; j < n; j++) {
             acc += sum(v,i,j);
          }
       }
       return acc;
    }

(c) int foo3(vector<int> v, int n) {
       if (n <= 1) {
          return sum(v,0,n);
       } else {
          return foo3(v, n-1) + foo3(v, n-2);
       }
    }

(d) int foo4(vector<int> v, int n) {
       if (n <= 1) {
          return sum(v,0,n);
       } else {
          return n*(n-1) + foo4(v, n-2);
       }
    }
```

(3) (24 pts.)

Say a string x is a *transpose* of a string y if y can be obtained from x by interchanging two different adjacent characters. For example, `sung` is a transpose of `snug`, but `sung` is not a transpose of `guns` because the `g` and the `s` are not adjacent. Nor is `sung` a transpose of `sang` even though they have edit-distace 1, and only words of the same length can be transposes. Finally, note that `meet` is a transpose of `mete` by switching the last two letters, but is *not* considered a transpose of itself by switching the two `e`'s, because those letters are the same.

Write in C++ a function `bool transpose(const string& x, const string& y)` which returns `true` if and only if `x` is a transpose of `y`. You may use C++ `string` library functions such as `substr` and `length`, and may use either `at` or the `operator[]` to refer to characters. Your code will have at least one loop and at least one `if(...)` test. *Finally*, you must write a comment explaining what it means if and when the loop executes to termination without being interrupted (say by a return statement), and what the `if(...)` being true means. END OF EXAM