COP 3530
Data Structures

Midsemester Exam

Name: _____

October 12, 1998

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 `CanSum` defined below. `CanSum` returns `true` if there are two integers in the array `A` that sum to exactly `K`. Ignore all syntax errors.

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

    return false;
}
```

   (a) What is the Big-Oh running time of this function?

   (b) If it takes 64 seconds to run `CanSum` on an array of 100000 elements, approximately how long will it take to run `CanSum` on an array of 50000 elements (assuming that low-order terms are negligible)?

   (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. (a) Write a function `Balance`, started below. `Balance` is passed a pointer to the root node of a binary tree and returns the difference between number of nodes the are left children and the number of nodes that are right children. In other words, if the tree has 8 nodes, of which one is the root, three are left children, and four are right children, then `Balance` returns -1. You may not add any global or static variables, or alter the function signature.

   (b) What is the running time of your algorithm?

```
int Balance( Node *T )
{
```

3. Function `SplitList` accepts an STL `list theList` and constructs two STL `lists` `evens` and `odds`. After `SplitList` returns, `evens` contains the items in `theList` that are in even positions (the first item is an odd position) in the original order, while `odds` contains the items in `oddList` that are in odd positions, in reverse order.

   For instance, if `theList` contains 3, 4, 1, 7, 5, then when the call returns, `evens` contains 4, 7, and `odds` contains 5, 1, 3. Complete the implementation of `SplitList` that is started below.

```
template <class Object>
void SplitList( const list<Object> & theList, list<Object> & evens,
                                               list<Object> & odds )
{
    list<Object>::const_iterator itr;    // to traverse theList
```

4. Write a complete generic queue class. You must support functions named `IsEmpty`, `MakeEmpty`, `Enqueue`, `Dequeue`, and `GetFront`. Your class should use a singly linked list implementation. You must define a `Node` class, either nested inside of `Queue` or separately. Make sure you signal errors. If you deem it necessary, you must write a destructor, copy constructor, and `operator=`. *You will receive no credit for an array implementation or an implementation that uses the STL* `list` *class.*