COSC 220 Final

Dr. Fred Park

Whittier College

Note: For full credit you must show all work. Incoherent work without logic or reason will not receive any credit whatsoever. You may use your class notes, any class code, and the textbook as well as the reference sites cplusplus.com and learncpp.com. No other resources allowed.

1. ~~(25 points)~~ If one would like to make pointers symmetrical in a linked list, the resulting data structure is called a *doubly linked list*. In the Beacons of Gondor linked list demo, we used a linked list to light each subsequent beacon starting at Minas Tirith and ending at Rohan. Create a doubly linked list that allows the beacons to be lit in both 1. the forward direction: starting at Minas Tirith and ending at Rohan as well as 2. the backward direction starting at Rohan and ending at Minas-Tirith. In the main part of the program, you will need to have the option of lighting from either end. Note: for each tower, you will need to now store 2 pointers.

2. ~~(25 points)~~ Rewrite the class implementation of the merge sort algorithm so that it sorts an array rather than a vector. Your function should use the prototype:
   `void sort(int array[], int n)`

3. (25 points) Suppose you have a vector of N elements, in which each element has a value in the inclusive range 1 to N-1. Given that there are N elements in the vector and only N-1 possible values to store in each slot, there must be at least 1 value duplicated in the vector. There may, of course, be many duplicated values, but you know there must be at least one by the **pigeonhole principle**. Same reasoning that on average, if you have 367 people in a room, then at least two will have the same birthday since on average there are 366 days in a year excluding leap years etc.

   Write a function
   `int findDuplicate(Vector<int> vec);` that takes a vector whose values are constrained to be

   in the 1 to N-1 range and returns one of the duplicated values. There is a catch though, your algorithm must have the following conditions:

   - It must run in $O(N)$ time.
   - It must use $O(1)$ additional space. i.e. it may use individual temporary variables but cannot allocate any additional array or vector storage. This also rules out recursive methods since space for stack frames etc. grow with recursion depth.
   - It may not change any values in the vector.

   A quadratic time solution is easy to obtain:

```
int findDuplicate(Vector<int> & vec) {
     for (int i = 0; i < vec.size(); i++) {
         for  (int j = 0; j < vec.size(); j++){
             if (vec[i] == vec[j]) return vec[i];
         }
}
error("Vector has no duplicates");
return -1;
}
```

   The hard part is optimizing the quadratic time solution so that it runs in linear time. Make sure to have your code output that it does indeed run in linear time.

4. (25 points) Write an outline/pseudo-code on how you would implement a stack using

   (a) an array

   (b) a linked list.

   You do not need to have C++ running code, just an outline/pseudo-code of how you would implement each. Include a diagram in each case.