Midterm Preview

… some samples of the types of questions you should expect …

20 Items -- 20 Points + 5 Points for Take-Home Portion = 25 Points Total

Select the best answer/response to each item below. Enter your answer on a Scantron sheet that is marked with your name and your SID.

The exam is <u>individual work</u>, <u>open books/notes</u>, <u>open e-books.</u> . Unless otherwise stated, wherever references are made to C++ code, you should place them in the context of the C++ implementations that were the subject of your CSE 330 lectures and labs.

Hand in: (1) Your Scantron sheet with your name and SID.
(2) These exam pages with your name and SID (ok to have written on them).

What is true about removing an element from a vector data structure?
(a) It is computationally cheaper to remove an element at a lower index than it is to remove an element at a higher index.
(b) Removing the element at highest index is always an O(1) operation.
(c) The vector data structure is designed to be a container of elements; it does not support element removal.
(d) Removing the highest index element involves shifting mySize-1 many elements one position to the right.

Q4: A vector data structure is implemented so that the two data members `int theSize` and `int myCapacity` will always remain properly synchronized. The implementations will make sure that ...
(a) The value of `theSize` will never be equal to the value of `myCapacity`.
(b) The value of `theSize` will always be less or equal `myCapacity`.
(c) The value of `theSize` will always be equal to `myCapacity`.
(d) The value of `myCapacity` will be increased when `theSize` has reached a value that is a constant amount larger than myCapacity.

Q6: If the list data structure did not maintain a data member `mySize` for the purpose of keeping track of the number of elements contained, which code fragment below could compute the size of a list?

(a) `iterator i = begin(); while (i < 0) i++; return i;`
(b) `return tail - 1;`
(c) `Node<T>* i = head->next; int k=0; while (i != 0) {k++; i = i->next;} return *i;`
(d) `Node<T>* i = head->next; int k=0; while (i != 0) {k++; i= i->next;} return k;`

Q18. What does the following code fragment compute for a given List mylst of integers? Assume that the list is not empty and elements are stored in ascending order of their values.

```
List<int>::iterator i1 =  mylst.begin();
List<int>::iterator i2 = mylst.end();

while (true) {
   if (i1 == i2) break;
   if (i1->next == i2) break;
   ++i1; ++i2;
}
if (i1 ==  i2)
  return *i1;
else
  return (*i1+*i2)/2;
```

(a) The middle element of the vector.
(b) The median value of the vector.
(c) The average between the smallest and largest value in the vector.
(d) This won't work.