CS 308 Data Structures

Spring 2002 - Dr. George Bebis

Exam 1

Duration: 1:00 - 2:15 pm

Name:

1. **True/False** (3 pts each) **To get credit, you must give brief reasons for your answers !!**

(1.1) **T F** Binary search is always faster than linear search.

(1.2) **T F** When an array is passed to a function, the function receives a copy of the array (call by value).

(1.3) **T F** Changes in the implementation of a class should not require changes in an application that uses the class.

(1.4) **T F** The running time of *RetrieveItem* (sorted lists) is *O(N)*

(1.5) **T F** An objective way to compare two algorithms is by comparing their execution (i.e., machine) times.

(1.6) **T F** Color images take up twice as much memory compared to gray-level images.

(1.7) **T F** An $O(logN)$ algorithm is slower than an $O(N)$ algorithm.

(1.8) **T F** The most appropriate structure to print a list of elements in reverse order is the Queue.

(1.9) **T F** The parameter to a copy constructor must be passed by reference.

(1.10) **T F** The running time of the program fragment shown below is $O(N)$

```
sum = 0;
for(i=0; i<N; i++) {
  if(i > j)
      sum = sum + 1;
  else {
    for(k=0; k<N; k++)
      sum = sum - 1;
  }
}
```

2. **Questions** (5 pts each)

(2.1) Analyze the running time of the function *InsertItem* shown below (sorted list). To get credit, you need to be as specific as possible.

```
template <class ItemType>
void SortedType<ItemType>::InsertItem(ItemType item)
{
 int location = 0;
 bool found;

 found = false;
 while( (location < length) && !found) {

   if(item > info[location])
     location++;
   else
     found = true;

 }

 for(int index = length; index > location; index--)
   info[index] = info[index - 1];
 info[location] = item;
 length++;
}
```

(2.2) What are the main differences between static and dynamic array allocation?

(2.3) Give the C++ statements for the dynamic allocation of an array with 3 rows and 5 columns. Draw a diagram that shows the structure of the dynamic array in memory.

(2.4) In programming assignment 1, you implemented a function that takes an image and *shrinks* it by a given factor. Describe in simple words how the *shrink* function works (no code). Assuming NxN images, give the running time of the function in terms of N, using big-O notation. Justify your answer.

(2.5) What are the differences between "call by value" and "call by reference" ?

(2.6) Demonstrate the binary search algorithm on the list (array-based) shown below. The element we want to retrieve is 55 (note that I am not asking you to write down the code; just include some figures that show the values of *first*, *last* and *mid* indices at each iteration).

0	12
1	31
2	44
3	54
4	96
5	100
6	200

3. **Code** (20 pts) Overload the assignment operator for the class *SortedType* (i.e., sorted linked list).

```
template<class ItemType>
class SortedType {
 public:
   SortedType();
   ˜SortedType();
   void MakeEmpty();
   bool IsFull() const;
   int LengthIs() const;
   void RetrieveItem(ItemType&, bool&);
   void InsertItem(ItemType);
   void DeleteItem(ItemType);
   void ResetList();
   bool IsLastItem() const;
   bool GetNextItem(ItemType&);
 private:
    int length;
    NodeType<ItemType> *listData;
    NodeType<ItemType> *currentPos;
};
```

4. **Code** (20 pts) Write a **client** function that merges two sorted lists using the following specification:

MergeLists(SortedType list1, SortedType list2, SortedType& result)

Function: Merges two sorted lists into one sorted list.

Precondition: list1 and list2 have been initialized.

Postconditions: result is a sorted list that contains all of the items from list1 and list2 (no duplicates)