EECE.3220: Data Structures

Spring 2017

Exam 2 Solution

1. (20 points, 5 points per part) ***<u>Multiple choice</u>***

For each of the multiple choice questions below, clearly indicate your response by circling or underlining the one choice you think best answers the question.

a. Say you want to design a matrix class (i.e., a two-dimensional table of numbers). Which of the following possible implementations requires you to write your own copy constructor and assignment operator, rather than just using the compiler-generated default versions?

A.
```
class Matrix {
…
private:
   double m[MAX_ROWS][MAX_COLS]; // Statically allocated
}                                //  2D array to hold matrix
```

B.
```
class Matrix {
…
private:
   int nr, nc;   // Number of rows/cols
   double **m;   // Pointer to dynamically allocated
}                //  2D array to hold matrix
```

C.
```
class Matrix {
…
private:
   double m[MAX_ROWS*MAX_COLS];  // Statically allocated
}                                //  1D array to hold matrix
```

D.
```
class Matrix {
…
private:
   Node *m;      // Pointer to linked list to hold matrix
}                // Assume each Node holds row #, col #, and
                 //   actual value in that matrix entry
```

i. Only A

ii. Only B

iii. A and C

iv. <u>B and D</u>

v. B, C, and D

1 (continued)
b. You are given the following function prototype and definition:

```
int f(int x, int y = 32);      // Prototype

int f(int x, int y) {          // Definition
    return y / x;
}
```

Which of the following function calls will result in the variable v being set to 4?

A. `v = f(8);`

B. `v = f(4);`

C. `v = f(10, 40);`

D. `v = f(2, 8);`

i. Only A

ii. Only B

iii. C and D

iv. B and C

v. <u>A, C, and D</u>

1 (continued)

c. Given two integers, `x` and `y`, if `x = 32` and `y = 20`, which of the following function calls will print an error message and terminate the program?

i. `assert(x > y);`

ii. ***`assert(x < y);`***

iii. `assert(x != 0);`

iv. None of the above

d. Which of the following statements accurately reflect your opinion(s)? Circle all that apply (but please don't waste too much time on this "question")!

i. "Are we done yet?"

ii. "I hope we get Program 4 before the end of the semester."

iii. "I hope we get assignment grades before the end of the semester."

iv. "I hope the rest of the exam is as easy as this question."

All of the above are "correct."

2. (40 points) ***Stacks and queues***
For each short program shown below, list the output exactly as it will appear on the screen. Be sure to clearly indicate spaces between characters when necessary.

You may use the available space to show your work as well as the output; just be sure to clearly mark where you show the output so that I can easily recognize your final answer.

Please refer to the additional handout provided with the exam for a description of `Stack` and `Queue` functions.

a. (14 points)

```
int main() {
   Stack st1;
   int i;
   int m[16] = { 0,   1, -2,  -3,
                -4,   5,  6,  -7,
                -8,  -9, 10, -11,
               -12, -13, 14,  15};

   for (i = 0; i < 16; i++) {
      if (m[i] >= 0)
         st1.push(i);
      else {
         cout << i << " ";
         i = i + 2;
      }
   }
   cout << "\n";

   while (!st1.empty()) {
      cout << st1.top() << " ";
      st1.pop();
   }
   cout << "\n";
   return 0;
}
```

OUTPUT:

```
2 7 11
15 14 10 6 5 1 0
```

2 (continued)
b. (12 points)

Please refer to the additional handout provided with the exam for a description of `Stack` and `Queue` functions.

```
int main() {
   Queue Qbert;
   int i;
   for (i = 10; i > 0; i--) {
      switch (i % 3) {
         case 0:
            if (!Qbert.empty())
               Qbert.enqueue(Qbert.front());
            break;
         case 1:
            Qbert.enqueue(i);
            break;
         case 2:
            Qbert.dequeue();
            break;
      }
   }

   while (!Qbert.empty()) {
      cout << Qbert.front() << "\n";
      Qbert.dequeue();
   }
   return 0;
}
```

OUTPUT:

```
10
4
7
1
```

2 (continued)
c. (14 points)

__**Please refer to the additional handout provided with the exam for a description of `Stack` and `Queue` functions.**__

```
int main() {
   Stack S;
   Queue Q;

   int list[10] = {3, 10, 0, -5, 9,
                   78, 6, 3220, 1, 1146};

   for (int i = 0; i < 10; i = i + 2)
      S.push(list[i]);

   while (!S.empty()) {
      cout << "S " << S.top() << "\n";
      Q.enqueue(list[S.top()]);
      S.pop();
   }

   while (!Q.empty()) {
      cout << "Q " << Q.front() << "\n";
      Q.dequeue();
   }
   return 0;
}
```

OUTPUT:

```
S 1
S 6
S 9
S 0
S 3
Q 10
Q 6
Q 1146
Q 3
Q -5
```

3. (40 points) ***Operator overloading; lists***

a. `AList operator -(const AList &rhs);`

This binary subtraction operator, used with static array-based lists, returns a new list which contains values that are only in the list to the left of the - operator, not in both the left and right.

For example, if `AList L1` holds {1, 2, 5, 9, 13} and `AList L2` holds {2, 5, 13}, the result of `L1 - L2` would be a list holding {1, 9}—in essence, the values in `L2` are removed from `L1`.

To simplify your solution, assume (1) both lists are sorted from lowest to highest values, and (2) the last value in the right hand list is greater than or equal to the last value in the left hand list.

```
AList AList::operator -(const AList &rhs) {
  AList result;          // Result of subtract operation
  int LHindex, RHindex;  // Index vars for left/right hand sides
  int i;                 // Index into result

  // Initialize variables
  LHindex = RHindex = i = 0;

  // Loop while LHindex/RHindex point to valid array locations
  while (LHindex < mySize && RHindex < rhs.mySize) {

    // Search RHS until you find value that matches or exceeds
    // current value in LHS, or run out of room in RHS
    while (rhs.myArray[RHindex] < myArray[LHindex] &&
              RHindex < rhs.mySize)
      RHindex++;

    // If value should be placed in result, add it
    if (RHindex < rhs.mySize &&
         myArray[LHindex] != rhs.myArray[RHindex])
      result.insert(myArray[LHindex++], i++);

    // Otherwise, if match found, skip to next value in LHS
    else if (RHindex < rhs.mySize)
      LHindex++;


    // NOTE: AS NOTED DURING THE EXAM, YOU SHOULD INCREMENT
    //   LHindex IN ALL CASES—FUNCTION COULD BE BETTER WRITTEN
  }
  return result;
}
```

3 (continued)

b. `bool operator ==(const DList &rhs);`

This binary comparison operator, used with dynamic array-based lists, returns true if the two lists (left- and right-hand-side) match and false otherwise. In order for two lists to match, the following conditions must be true:

- The lists must be the same size (but not necessarily the same capacity)
- The lists must store the same set of values

Students were responsible for writing bold, underlined, italicized code.

```
bool DList::operator ==(const DList &rhs) {
   int i;       // Index

   // If sizes don't match, lists don't match—return appropriate
   //    value

   if (mySize != rhs.mySize)
      return false;

   // If sizes do match, check to see if array contents match
   // If mismatch found, can end function at that point
   else {
      for (i = 0; i < mySize; i++) {
         if (myArrayPtr[i] != rhs.myArrayPtr[i])
            return false;
      }
   }

   // If sizes and array contents match, lists match—return
   //    appropriate value
   return true;
}
```

3 (continued)

c. `LList & operator ~();`

This unary operator, used with linked lists, reverses the node order of the calling object. For example, if `LList L1` contains nodes `1 -> 2 -> 3 -> 4`, the statement `~L1;` would change the list so the nodes were in the order `4 -> 3 -> 2 -> 1`.

The solution requires you to keep three pointers: the current node, its original predecessor, and its original successor. Your function should visit all nodes in the list. For each node, (1) make the node point to its original predecessor, then (2) move all three pointers ahead by one node.

Assume—as with the linked list covered in class—that the `insert()` function adds nodes so they're ordered from lowest to highest value, so you can't use that function in your solution.

Students were responsible for writing bold, underlined, italicized code.

```
LList & LList::operator~() {
   Node *pred;       // Original predecessor of current node
   Node *cur;        // Current node
   Node *succ;       // Original successor of current node

   // Make sure list isn't empty
   if (!isEmpty()) {
      // Initialize node pointers
      pred = NULL;
      cur = first;
      succ = first->next;

      // Visit & reorder nodes while there's a non-NULL successor

      while (succ != NULL) {
         cur->next = pred;
         pred = cur;
         cur = succ;
         succ = succ->next;
      }

      // Make sure old last node points to its new successor, and
      //   "first" pointer points to new first node
      cur->next = pred;
      first = cur;
   }

   // Return reference to calling object
   return *this;
}
```