EECE.3220: Data Structures

Spring 2017

Exam 1 Solution

1. (20 points, 5 points per part) ***Multiple choice***

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.

NOTE: All parts of this question refer to the class `E1Class`, which is defined on the extra sheet included with the exam.

a. Which of the following statements is a valid declaration for an object of type `E1Class`?

A. `E1Class ec1;`

B. `E1Class ec2(1, 2);`

C. `E1Class ec3(3, 4, 5.6);`

D. `E1Class::ec4;`

i. Only A

ii. Only B

iii. A and C

iv. B and D

v. A, B, and C

b. Which of the following statements uses the correct syntax to call the "get" function `readVars()`, assuming variables `int x, y` and `double z` have been declared and the object `E1Class ec` has been properly initialized?

i. `E1Class.readVars(x, y, z);`

ii. ***`ec.readVars(x, y, z);`***

iii. `ec.readVars(&x, &y, &z);`

iv. `readVars(ec, x, y, z);`

v. None of the above

c. Assuming the "set" function `setDouble()` changes the value of the data member `var3` inside an `E1Class` object, which of the following statements will set `var3` inside the object `ec` to 3.4?

A. `ec.var3 = 3.4;`

B. `ec.setDouble(3.4);`

C. `ec.E1Class(1, 2, 3.4);`

D. `E1Class::setDouble(3.4);`

i. Only A

ii. ***Only B***

iii. A and B

iv. C and D

v. A, B, and C

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. "This course is moving too quickly."

ii. "This course is moving too slowly."

iii. "I've attended very few lectures, so I don't really know what the pace of the course is."

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

All of the above are "correct."

2. (30 points) ***C++ input/output***

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.

a. (15 points)

```
int main() {
    int x = 5;
    double d1, d2, d3, d4;
    d1 = 40.0;
    d2 = d1 / (x * 10);          d2 = 40.0 / 50 = 0.8
    d3 = d1 - d2;                d3 = 40.0 - 0.8 = 39.2
    d4 = d3 / 1000;              d4 = 0.0392

    cout << d1 << " " << d2 << " " << d3 << " " << d4 << endl;
    cout << fixed << showpoint;
    cout << d4 << " ";
    cout << setprecision(3);
    cout << d3 << " " << d2 << " ";
    cout << setprecision(1);
    cout << d1 << "\n";

    return 0;
}
```

Notes:

- The first line of output uses default settings, meaning (1) trailing 0s are not shown after the decimal point, and (2) values that have a fractional part = 0 are printed as integers.
- The second `cout` statement forces the decimal point to be shown; until the precision is changed, the default precision of 6 will be used, which explains why `d4` is printed with 6 digits after the decimal point.
- Since precision settings are sticky, changing the precision to 3 forces both `d3` and `d2` to be printed with a precision of 3.

OUTPUT:

```
40 0.8 39.2 0.0392
0.039200 39.200 0.800 40.0
```

2 (continued)
b. (15 points)

For this program, assume the user inputs the two lines below. The digit `'1'` is the first character the user types. There is one space (`' '`) between `1.23` and `4.56`, and one space between `7.89` and `10.1112`. Assume each line ends with a newline character (`'\n'`).

You must determine how the program handles this input and then print the appropriate results. Note that the program may not read all characters on the input lines, but no input statement fails to read data—an appropriate value is assigned to every variable.

```
     1.23 4.56
     7.89 10.1112

int main() {
    int i1, i2;
    double d1, d2;
    char c1, c2;
    char buf[10];

    cin >> i1 >> d1 >> c1 >> d2;    i1 = 1, d1 = 0.23, c1 = '4',
                                    d2 = 0.56
    cin.ignore(2);   Skips '\n' at end of first line, '7' at
                     start of second
    cin.get(c2);     c2 = '.'
    cin >> i2;       i2 = 89
    cin.getline(buf, 10);      Reads all remaining characters on
                               second line: " 10.1112\n" (newline
                               not shown above, but problem spec
                               does say each line ends with '\n')

    cout << i1 << " " << i2 << endl;
    cout << d1 << " " << d2 << endl;
    cout << c1 << c2 << endl;
    cout << buf << endl;
    return 0;
}
```

OUTPUT:

```
1 89
0.23 0.56
4.
 10.1112
```

3. (30 points) ***Algorithmic complexity***

For each function in this problem, determine (a) an equation for the worst-case computing time T(n) (expressed as a function of *n, i.e.* 2n + 4) and (b) the order of magnitude (expressed using big O notation, *i.e.* O(n)). Note that:

- Each executable line of code is numbered so you can refer to it by number if necessary.
- A `for` loop may be treated as a single statement, not three separate statements.

The number of times each line is executed is shown to the right of that line in red.

a. (15 points)

```
int functionA(int n){
1   int total = 1;                                  1
2   for (int i = 1; i <= n; i++) {                  n + 1
3       for (int j = 1; j <= n; j++) {              n*(n+1)
4           for (int k = 1; k <= n; k++) {          n*n*(n+1)
5               if (i == j && j == k)               n*n*n
6                   total = total * i;              n
            }
        }
    }
7   return total;                                   1
}
```

T(n) $= 1 + (n + 1) + n*(n+1) + n*n*(n+1) + n*n*n + n + 1$
$= 1 + n + 1 + n^2 + n + n^3 + n^2 + n^3 + n + 1$
$= \underline{2n^3 + 2n^2 + 3n + 3}$
$= \underline{O(n^3)}$

3 (continued)

For each function in this problem, determine (a) an equation for the worst-case computing time T(n) (expressed as a function of *n, i.e.* 2n + 4) and (b) the order of magnitude (expressed using big O notation, *i.e.* O(n)). Note that:

- Each executable line of code is numbered so you can refer to it by number if necessary.
- A `for` loop may be treated as a single statement, not three separate statements.

The number of times each line is executed is shown to the right of that line in red.

b. (15 points)

```
int functionB(int n){
1   int i = n;                  1
2   int r = 0;                  1
2   if (n < 2)                  1
3       r = 1;                  1 (if condition is true)
    else {
4       while (i > 0) {         log2n + 2
5           r = r + r;          log2n + 1
6           i = i / 2;          log2n + 1
        }
    }
7   return r;                   1
}
```

Notes:

- The worst-case path through the function is the "else" case, so that path should be the basis for your worst-case analysis.
- As one of you pointed out near the end of the exam, the `while` loop is pointless, since `r` will always be 0 regardless of the value of `n`. That's what happens when I mismanage my time and finish writing an exam around 2:00 in the morning.

T(n) $= 1 + 1 + 1 + \log_2 n + 2 + \log_2 n + 1 + \log_2 n + 1 + 1$

$= \underline{3 \log_2 n + 8}$

$= \underline{O(\log_2 n)}$

4. (20 points) ***Structures and functions***

For each part of this problem, you are given a short function to complete. **CHOOSE EITHER OF THE TWO PARTS** and fill in the spaces provided with appropriate code.

You may complete both parts for up to 10 points of extra credit, but must clearly indicate which part is the extra one—I will assume it is part (b) if you mark neither of them.

Remember, you must write all code required to make each function work as described—**do not assume you can simply fill in the blank lines and get full credit.**

Also, remember that if examples are provided, each example is only applicable in one specific case—**it does not cover all possible results of using that function.**

In order to allow plenty of space to solve each problem, this page is essentially just a "cover sheet" for Question 4—**the actual problems start on the next page**.

Each of these functions works with two structures. **You can find the structure definitions on the extra sheet provided with the exam.**

4 (continued)

a. `void QbyQavg(Exam &avg, Exam elist[], int n);`

This function takes an array `elist` that contains `n` `Exam` structures, computes the question-by-question average for each question, and stores that information in the `Exam` structure referenced by `avg`. The function should assign values to that `avg` structure as follows:

- The `name` field within `avg` should be set to `"Average"`.
- The `qlist` array in `avg` should be set as follows:
  - The `max` field in each `qlist` entry in `avg` is the same as in all entries of `elist`. (For example, `qlist[0].max` will be the same in all `Exam` structures.)
  - The `score` field in each `qlist` entry in `avg` should be the average of all corresponding `score` fields in `elist`.

Students had to write bold, underlined, italicized code.

```
void QbyQavg(Exam &avg, Exam elist[], int n) {
    int i, j;        // Loop indexes

    // Set name within avg
    avg.name = "Average";

    // Compute question-by-question average
    for (j = 0; j < 4; i++) {
        avg.qlist[j].max = elist[0].qlist[j].max;
        avg.qlist[j].score = 0;
        for (i = 0; i < n; j++)
            avg.qlist[j].score += elist[i].qlist[j].score;
        avg.qlist[j].score /= n;
    }

}
```

4 (continued)
b. `void gradeDist(int dist[5], Exam elist[], int n);`

This function takes an array `elist` that contains `n Exam` structures and determines the distribution of total grades represented by those structures.

The distribution should be stored in the array `dist[]`, with `dist[0]` representing the number of scores between 90-100, `dist[1]` scores between 80-89, `dist[2]` scores between 70-79, `dist[3]` scores between 60-69, and `dist[4]` scores below 60.

Students had to write bold, underlined, italicized code.

```
void gradeDist(int dist[5], Exam elist[], int n) {
    int i, j;        // Loop indexes
    int total;       // Total score on a given exam

    // Clear all entries in dist[] array
    for (i = 0; i < 5; i++)
        dist[i] = 0;

    // Compute total score for each Exam and count it in
    //   appropriate entry in dist[]

    for (i = 0; i < n; i++) {

        // Find total score for current exam
        total = 0;
        for (j = 0; j < 4; j++)
            total += elist[i].qlist[j].score;

        // Test score to determine which dist[] entry to update
        if (total >= 90)       dist[0]++;
        else if (total >= 80)  dist[1]++;
        else if (total >= 70)  dist[2]++;
        else if (total >= 60)  dist[3]++;
        else                   dist[4]++;
    }
}
```