EECE.3220: Data Structures

Fall 2019

Exam 1 Solution

1. (24 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. (12 points)

```
int main() {
   int iv1 = 6;
   int iv2 = 36;
   double dval = 103.2019;

   cout << iv1 << endl << iv2;
   cout << setprecision(4) << dval << '\n';

   cout << fixed << "Now, iv1 = "
          << iv1 << ", iv2 = " << iv2
          << ",\ndval = " << setprecision(3)
          << dval << endl;
   return 0;
}
```

At this point, fixed format isn't specified, so precision = # significant figures Also, lack of newline means dval (103.2) prints next to iv2 (36)

Now, precision = # digits after decimal point

OUTPUT:

```
6
36103.2
Now, iv1 = 6, iv2 = 36,
dval = 103.202
```

1 (continued)
b. (12 points)

For this program, assume the user inputs the three lines below. The letter `'E'` in `Exam` is the first character the user types. There is one space (`' '`) between `Exam` and `1` on the first line, one space between `2-4` and `PM` on the second line, and no spaces on the third line. 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.

```
Exam 1
2-4 PM
10.3.2019
```

```
int main() {
   int i1, i2;
   double d1, d2;
   char c1, c2;
   string str1, str2;

   cin.get(c1);
   cin.ignore(1);
   cin >> str1 >> i1 >> d1 >> i2 >> c2;
   getline(cin, str2);    // Slightly different getline syntax to
                          //  read line into string object
   cin >> d2;

   cout << i1 << ", " << i2 << '\n';
   cout << d1 << ", " << d2 << '\n';
   cout << c1 << ", " << c2 << '\n';
   cout << str1 << ", " << str2 << '\n';

   return 0;
}
```

Solution: *This program handles the input as follows:*

- *`cin.get(c1)` reads first character into `c1` → `c1 = 'E'`*
- *`cin.ignore(1)` skips a single character (`'x'`)*
- *Next line reads a string, an int, a double, another int, and a non-space character:*
  - *`str1 = "am"`*
  - *`i1 = 1`*
  - *`d1 = 2`*
  - *`i2 = -4`*
  - *`c2 = 'P'`*
- *`getline(cin, str2)` reads the rest of the line into `str2` → `str2 = "M"`*
- *`cin >> d2` reads the next real number into `d2` → `d2 = 10.3`*

1b. (continued) So, the output is as follows:

```
1, -4
2, 10.3
E, P
am, M
```

2. (12 points) ***Functions***

For the 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.

```
int f1(int &arg1, int arg2) {
   arg1 = arg1 * 2;
   arg2 = arg2 / 2;
   return arg1 + arg2;
}
```

arg1 is a reference argument, so function changes variable passed as first argument. arg2 is passed by value, so it's unchanged outside fn.

```
int f2(int *arg3, int arg4) {
   *arg3 = arg4 + 5;
   arg4 = arg4 - 5;
   return *arg3 + arg4;
}
```

arg3 is a pointer, so so function changes variable passed as first argument. arg4 is passed by value, so it's unchanged outside fn.

```
int f3(int &arg5, int &arg6) {
   arg5 = arg6;
   arg6 = arg5 / 4;
   return arg5 + arg6;
}
```

Both arguments are reference arguments, so function changes both variables passed to it

```
int main() {
   int v1, v2(5), v3(8);
   v1 = f1(v2, v3);
   cout << v1 << ' ' << v2 << ' ' << v3 << '\n';

   v2 = f2(&v3, v1);
   cout << v1 << ' ' << v2 << ' ' << v3 << '\n';

   v3 = f3(v1, v2);
   cout << v1 << ' ' << v2 << ' ' << v3 << '\n';

   return 0;
}
```

*v2 = v2 * 2 = 10, v3 unchanged*
v1 = v2 + v3/2 = 14

v3 = v1 + 5 = 19, v1 unchanged
v2 = v3 + (v1 – 5) = 28

v1 = v2 = 28
v2 = v1 / 4 = 7
v3 = v1 + v2 = 35

OUTPUT:

```
14 10 8
14 28 19
28 7 35
```

3. (20 points) ***Strings***

a. (10 points) For the 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.

```
int main() {
   string s1, s2("exam"), s3("October");

   if (s1.empty() && s2.empty()) {        Condition is false since
      s1 = s3;                            s2 is non-empty
      s2 = "test";
   }
   else if (s1.length() < 5) {            s1.length() = 0, so
      s1 = "test";                        condition is true and
   }                                      s1 is set to "test"
   else
      s3 = "test";

   cout << s1 << ' ' << s2 << ' ' << s3 << '\n';

   s1[2] = s2.at(1);              Changes 's' to 'x' → s1 = "text"

   s2 += "ple";                   s2 = "exam" + "ple" = "example"

   s3 = s2.substr(2, 2) + s3.substr(4);  s3 = "am" + "ber"
                                            = "amber"

   cout << s1 << ' ' << s2 << ' ' << s3 << '\n';

   return 0;
}
```

OUTPUT:

```
test exam October
text example amber
```

3 (continued)
b. (10 points) Complete the function described below:

```
void removeDuplicates(string &str);
```

Removes all instances of consecutive, repeated characters from the argument `str`, leaving just one of each character in the string. For example:

- If `string s1 = "aa bb cc dd"`, after calling `removeDuplicates(s1)`, `s1 = "a b c d"`
- If `string s2 = "no repeats"`, after calling `removeDuplicates(s2)`, `s2 = "no repeats"`
- If `string s3 = "1223334444"`, after calling `removeDuplicates(s3)`, `s3 = "1234"`

The comments in the function provide hints about one potential implementation of this function; if you have a different solution, feel free to ignore the comments.

Other solutions may be valid

```
void removeDuplicates(string &str) {
   unsigned i;            // Loop index
   string tmp;            // Temporary string

   // Identify repeated characters in str and only copy one of
   //    each to tmp
   i = 1;
   while (i < str.length()) {
      if (str.at(i) != str.at(i - 1))
         tmp += str.at(i - 1);
      i++;
   }
   tmp += str.at(str.length() - 1);

   // Copy temporary string to str before end of function
   str = tmp;
}
```

4. (20 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.

a. (10 points)

```
int functionA(int n){
    int i, j;
1   int total = 1;                    1
2   int count = 0;                    1
3   for (i = 0; i <= n; i++) {        n + 2
4       count++;                      n + 1
5       j = i;                        n + 1
6       while (j > 0) {               n*(n+1)/2 + 1 = n²/2 + n/2 + 1
7           total = total * 2;        n*(n+1)/2 = n²/2 + n/2
8           j--;                      n*(n+1)/2 = n²/2 + n/2
        }
    }
9   return total;                     1
}
```

Worst case analysis: The nested while loop is the trickiest part of this problem, which I graded leniently. Here's how we get the formula n*(n+1)/2 for the total number of while loop iterations:

- The number of while loop iterations matches the index value, i, for the for loop—the function sets j = i outside the while loop, then count down until j = 0.
- i goes from 0 to n. So, when j = 0, the while loop has 0 iterations. When j = 1, the while loop has 1 iteration. That pattern continues, with the last for loop iteration containing a while loop with n iterations.
- The total number of iterations for the while loop can therefore be expressed by the sum: 0 + 1 + 2 + … + n, which can itself be expressed by the formula n*(n+1)/2.

Therefore, $T(n) = 1 + 1 + (n + 2) + (n + 1) + (n + 1) + (n^2/2 + n/2 + 1) + (n^2/2 + n/2) + (n^2/2 + n/2) + 1$
$= 3n^2/2 + 9n/2 + 8$

The order of magnitude for the worst-case execution time is therefore $\mathbf{O(n^2)}$.

4 (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.

b. (10 points)

```
void functionB(int arr[], int n){
    int i, j;
1   if (n >= 20) {                          1
2       for (i = 0; i < 20; i++)            21
3           arr[i]++;                       20
    }
4   if (n >= 30) {                          1
5       for (i = 0; i < 30; i++) {          31
6           for (j = i; j < 30; j++) {      32 * 31 / 2 = 496
7               arr[i] += arr[j];           31 * 30 / 2 = 465
            }
        }
    }
8   if (n >= 40) {                          1
9       for (i = 39; i >= 0; i--)           41
10          arr[i] = arr[i] * 2;            40
    }
}
```

Worst case analysis: I didn't plan this, but I used the same loop structure on parts (a) and (b). The number of iterations for the inner for loop starting on line 6 depends on the index of the outer loop ... just like the nested while loop in the previous problem. So, n*(n+1)/2 gives you the total number of iterations. For line 6, n = 31, as you're adding values from 1 to 31 (line 6 executes 31 times when i = 0). For line 7, n = 30. That's where 496 and 465 come from.

In all cases, though, note that the number of iterations is independent of n—every loop uses a constant as a boundary. The worst case would be for n to be greater than or equal to 40, making all three conditions (lines 1, 4, and 8) true and causing all lines of the function to execute. So:

```
T(n) = 1 + 21 + 20 + 1 + 31 + 496 + 465 + 1 + 41 + 40 = 1117
```

and the order of magnitude for that worst-case execution time is **O(1).**

5. (24 points, 4 points each) ***Multiple choice: classes, dynamic allocation, vectors***

For each of the multiple choice questions below, clearly indicate your response(s) by circling or underlining all choices you think best answer the question.

Parts a, b, and c refer to the following class definition:

```
class MyClass {
public:
   MyClass();
   MyClass(int i, string s);
   void setMem(int arg1, int arg2, string arg3);
   void display(ostream &out);
private:
   int mem1, mem2;
   string mem3;
};
```

a. Which of the following statements is a valid declaration for an object of type `MyClass`? **This question has at least one correct answer, but may have more than one correct answer! Circle ALL choices that correctly answer the question.**

i. ***`MyClass mc1;`***

ii. ***`MyClass mc2(1, "two");`***

iii. `MyClass mc3(3, 4, "five");` ***Parameterized constructor only takes 2 arguments***

iv. `MyClass mc4.mem1 = 3;` ***All sorts of wrong … can't declare an object and immediately access a data member, and that data member is private anyway***

v. `MyClass(0, "zero");` ***Doesn't actually declare anything—there's no object name here, just the type name***

5 (continued)

Part b uses the same class definition as part a:

```
class MyClass {
public:
   MyClass();
   MyClass(int i, string s);
   void setMem(int arg1, int arg2, string arg3);
   void display(ostream &out);
private:
   int mem1, mem2;
   string mem3;
};
```

b. Which of the following choices represents a valid implementation of a `MyClass` constructor?

This question has at least one correct answer, but may have more than one correct answer! Circle ALL choices that correctly answer the question.

i.
```
MyClass() {
   mem1 = 0;
   mem2 = 1;
   mem3 = "blank";
}
```
Missing MyClass:: before fn. name

ii.
```
MyClass::MyClass() : mem1(10), mem2(3), mem3("Exam 1")
{}
```

iii.
```
MyClass::MyClass(int i, string s) : mem1(i), mem2(i),
   mem3(s)
{}
```

iv.
```
MyClass::setMem(int arg1, int arg2, string arg3) {
   mem1 = arg1;
   mem2 = arg2;
   mem3 = arg3;
}
```
Perfectly good setMem() definition, but that function isn't a constructor

5 (continued)

Part c uses two class definitions:

```
class MyClass {
public:
   MyClass();
   MyClass(int i, string s);
   void setMem(int arg1, int arg2, string arg3);
   void display(ostream &out);
private:
   int mem1, mem2;
   string mem3;
};

class NewClass {
public:
   NewClass();
   NewClass(int i1, int i2, string s1, string s2);
   void someFunction(MyClass &randomReference);
private:
   MyClass x, y;
}
```

c. Which of the following represents a valid implementation of the `NewClass` parameterized constructor? **This question has exactly one correct answer.**

i.
```
NewClass::NewClass(int i1, int i2, string s1, string s2) {
   x = (i1, s1);
   y = (i2, s2);
}
```

ii.
```
NewClass::NewClass(int i1, int i2, string s1, string s2) {
   x.mem1 = x.mem2 = i1;
   x.mem3 = s1;
   y.mem1 = y.mem2 = i2;
   y.mem3 = s2;
}
```

iii.
```
NewClass::NewClass(int i1, int i2, string s1, string s2) :
   x(i1, s1), y(i2, s2) {}
```

iv.
```
NewClass::NewClass(int i1, int i2, string s1, string s2) {
   someFunction(x);
   someFunction(y);
}
```

5 (continued)

d. Which of the following choices dynamically allocates an array of doubles in which `N` represents the array size and `M` represents the initial value assigned to each element in the array? **This question has exactly one correct answer.**

i. `double *arr = new double(N, M);`

ii. `double *arr = new double[N, M];`

iii. `double arr[N] = { M };`

iv.
```
double *arr = new double[N];
for (unsigned i = 0; i < N; i++)
   arr[i] = M;
```

e. Which of the following choices creates a vector of integers containing 5 elements and assigns the value 3220 to every element? **This question has at least one correct answer, but may have more than one correct answer! Circle ALL choices that correctly answer the question.**

i. `vector <int> v1(5) = 3220;`

ii.
```
vector <int> v2(5, 3220);
```

iii.
```
vector <int> v3 = {3220, 3220, 3220, 3220, 3220};
```

iv.
```
vector <int> v4;
for (unsigned i = 0; i < 5; i++)
   v4.push_back(3220);
```

f. 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 next exam is as easy as this question."

All of the above are "correct."

6. (10 points) ***EXTRA CREDIT***

Complete the function below, which returns a string formed by combining all substrings within `s1` that start with the character `c` and have length `len`. When building the new string, add a space after each added substring. For example:

- `buildStr("testing string", 't', 3)` returns `"tes tin tri"`
- `buildStr("accident", 'c', 2)` returns `"cc ci"`
- `buildStr("EECE.3220", 'x', 5)` returns `""` (empty string)

```
string buildStr(string s1, char c, int len) {
   unsigned i;            // Character position within string
   string result;         // Final result

   // GO THROUGH s1 AND FIND SUBSTRINGS TO CONCATENATE
   // Test all characters
   i = 0;
   while (i < s1.length()) {

      // If match is found, add substring of length len
      //   and space to result
      if (s1.at(i) == c) {
         if (!result.empty())
             result += ' ';
         result += s1.substr(i, len);
      }
      i++;
   }

   return result;
}
```