16.216: ECE Application Programming

Spring 2012

Exam 2 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 choice you think best answers the question.

a. You are given the following short program:

```
int i;
for (i = 0; i < 20; i++) {
   if (i < 10)
      continue;

   if ((i % 2) == 1)
      break;
}
```

How many iterations will this loop execute?

i. 10

ii. 11

iii. 12

iv. 20

v. An infinite number—the loop never ends

b. You are given the following short program:

```
void main() {
   int x, y;
   int *p1, *p2;

   scanf("%d", &x);
   y = x;
   p1 = &x;
   p2 = p1;

   x = x + 2;     // REPLACE THIS STATEMENT
   printf("%d", x);
}
```

Which of the following statements could replace the underlined statement and always generate the exact same program output?

i. `y = y + 2;`

ii. `p1 = p1 + 2;`

iii. ***`*p1 = *p2 + 2;`***

iv. None of the above

1 (cont.)

c. Given the following code snippet:

```
int x = 20;
int i = 0;              // START OF CODE TO BE REPLACED
while (i <= 9) {
   x = x * 2;
   i++;
}                       // END OF CODE TO BE REPLACED
```

Which of the following choices can replace the code indicated by the comments and produce the exact same value for x? Assume x is always initialized to 20.

i.
```
for (i = 9; i > 0; i--) {
   x = x * 2;
}
```

ii.
```
for (x = 0; x <= 9; x++) {
   x = x * 2;
}
```

iii.
```
for (i = 10; i <= 100; i += 10) {
   x = x + x;
}
```

iv.
```
for (i = 0; i < 9; i++) {
   x = x * 2;
}
```

d. Which of the following "functions" would you find most useful? Circle all that apply *(and please don't waste too much time on this "question").*

i.
```
void set_date(int month, int day, int year);
set_date(5, 18, 2012);   // In case you're wondering, final
                         //   exams end on 5/17/2012
```

ii.
```
copy_exam_solution(&my_exam);
```

iii.
```
drop_lowest_five_program_scores(&my_grades);
```

iv.
```
set_overall_GPA(4.0);
graduate_now();
```

Any of the above is "correct," but (i) is the only one that would help both students and teachers.

2. (40 points) ***Functions and pointers***

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. (14 points)

```
void main() {
     int d1, d2;
     int *p1, *p2;

     d1 = 16;
     d2 = 216;
     p1 = &d2;
     p2 = &d1;
     d2 = *p2 - 15;
     *p2 = *p1 + 2;
     p1 = p2;
     d1 = d2 * 4;
     *p1 = *p2 + 15;

     printf("%d %d %d %d\n", d1, d2, *p1, *p2);
}
```

Solution:

- After first statements, `d1 = 16` and `d2 = 216`, `p1` points to `d2`, and `p2` points to `d1`.
- `d2 = *p2 - 15;` → `d2 = d1 – 15 = 16 – 15 = 1`
- `*p2 = *p1 + 2;` → `d1 = d2 + 2 = 1 + 2 = 3`
- `p1 = p2;` → `p1` now points to `d1`
- `d1 = d2 * 4;` → `d1 = 1 * 4 = 4`
- `*p1 = *p2 + 15;` → `d1 = d1 + 15 = 4 + 15 = 19`

FINAL OUTPUT:

```
19 1 19 19
```

2 (cont.)

b. (14 points)

```
double max(double d1, double d2) {
     if (d1 > d2)
          return d1;
     else
          return d2;
}

double min(double d1, double d2) {
     if (d1 < d2)
          return d1;
     else
          return d2;
}

void main() {
     double x = 1.3;
     double y = 2.4;
     double z = 3.5;

     double a, b, c;

     a = min(x, y);
     b = max(a, z);
     c = min(b, y);
     x = max(y, c);
     y = min(a, x);
     z = max(x, y);

     /* Each format specifier contains a "one" followed by a
          lowercase 'L'  */
     printf("%.1lf %.1lf %.1lf\n", x, y, z);
     printf("%.1lf %.1lf %.1lf\n", a, b, c);
}
```

Solution:

- a = min(x, y) = min(1.3, 2.4) = 1.3
- b = max(a, z) = max(1.3, 3.5) = 3.5
- c = min(b, y) = min(3.5, 2.4) = 2.4
- x = max(y, c) = max(2.4, 2.4) = 2.4
- y = min(a, x) = min(1.3, 2.4) = 1.3
- z = max(x, y) = max(2.4, 1.3) = 2.4

FINAL OUTPUT:

```
2.4 1.3 2.4
1.3 3.5 2.4
```

c. (12 points)

```
void badSwap(int *x, int y) {
     int temp;
     temp = *x;
     *x = y;
     y = temp;
}

void badSwap_2(int x, int *y) {
     int temp;
     temp = x;
     x = *y;
     *y = temp;
}

void main() {
     int a = 1;
     int b = 2;
     int c = 3;
     int d = 4;

     badSwap(&a, b);
     badSwap_2(c, &d);
     badSwap(&b, c);
     badSwap_2(a, &d);

     printf("%d %d %d %d\n", a, b, c, d);
}
```

Solution: The key part of this problem is that each of the function calls only modifies one value in the main program—whatever variable is passed by address. Therefore:

- `badSwap(&a, b);` → `a = b = 2, b remains unchanged`
- `badSwap_2(c, &d);` → `d = c = 3, c remains unchanged`
- `badSwap(&b, c);` → `b = c = 3, c remains unchanged`
- `badSwap_2(a, &d);` → `d = a = 2, a remains unchanged`

FINAL OUTPUT:

```
2 3 3 2
```

3. (40 points, 20 per part) ***Loops and switch statements***

a. Complete the program below so that it prompts for and reads a character followed by two integers (`in1` and `in2`), then executes one of the following, based on the character value:

- `'S'`: Calculate the sum of the numbers.
- `'L'` or `'G'`: Calculate the larger of the two numbers.
- Any other input character: Operation result is 0

After any of these calculations, print the result as shown. Examples (with user input underlined):

```
Enter cmd in1 in2: S 32 500
Result = 532
Enter cmd in1 in2: L 5 17
Result = 17
```

```
Enter cmd in1 in2: M 15 2
Result = 0 (erroneously written as 1 in actual exam)
Enter cmd in1 in2: m 15 2
Result = 0
```

Code in bold, italics was to be filled in by students

```
void main() {
      int in1, in2, result;       // Input values and final result
      char cmd;                   // Input command

      printf("Enter cmd in1 in2: ");        // Prompt for and
      scanf("%c %d %d", &cmd, &in1, &in2);  //   read inputs

      // Use value of cmd to determine which operation to execute
      switch (cmd) {

      case 'S':                   // Calculate sum
            result = in1 + in2;
            break;

      case 'L':                   // Calculate larger of 2 values
      case 'G':
            if (in1 > in2)
                  result = in1;
            else
                  result = in2;
            break;

      default:
            result = 0;
      }

      printf("Result = %d\n", result); // Print final result
}
```

3 (cont.)

b. Complete the program below so that it repeatedly reads a value (`limit`), then calculates and prints all numbers in the Fibonacci sequence that are less than `limit`. The first two Fibonacci numbers are 0 and 1, and every subsequent value in the sequence is the sum of the two numbers that precede it: ***0, 1, 1, 2, 3, 5, 8, 13, 21, 34 …***

   The program should end when the user enters a limit that would prevent you from printing at least the first three values in the sequence. Examples (with user input underlined):

```
Enter limit: 5
0 1 1 2 3
Enter limit: 40
0 1 1 2 3 5 8 13 21 34
Enter limit: 100
0 1 1 2 3 5 8 13 21 34 55 89
Enter limit: -1
```

Code in bold, italics was to be filled in by students

```
void main() {
      int limit;       // Upper limit of sequence
      int num1, num2;  // Values used to calculate sequence

      // Loop that ensures user can repeatedly enter new limit
      while (1) {      // Loop can be infinite, since the break
                       //    below exits it

            printf("\nEnter limit: ");  // Prompt for and read limit
            scanf("%d", &limit);

            // Exit if limit is too low to print at least 3 values
            if (limit < 2)        // Could use return, not break,
                  break;          //   since ending loop ends program

            // Initialize num1 and num2 and print first 2 values
            num1 = 0;
            num2 = 1;
            printf("0 1 ");

            // Loop that keeps calculating Fibonacci numbers as long as
            //   next value is less than limit
            while (num1 + num2 < limit) {
                  printf("%d ", num1 + num2);      // Print next value

                  // Update num1 and num2
                  num2 = num1 + num2;         // num2 = most recent value
                  num1 = num2 – num1;         // num1 = old value of num2
            }
      }
}
```

To see why the equation for `num1` *works, let* `num1′` *and* `num2′` *be the new values of* `num1` *and* `num2` *calculated in each loop iteration. Once we overwrite* `num2`*, to get its old value, we have to work with the equation for the new value.*

Since `num2′ = num1 + num2`*, it follows that* `num2 = num2′ – num1 = num1′`

3 (cont.)

c. The program below will generate a random value between 1 and 100. The user has a certain number of attempts—which is specified by the user—to correctly guess the value. The program should end when either:

- The user correctly guesses the random value, or
- The user has reached the maximum number of guesses.

Note that, for each guess, the program prints the guess number—the first guess is #1, the second is #2, and so on. Complete this program so that it behaves as specified. Examples:

```
Enter max # guesses: 2
Enter guess 1: 10
Wrong
Enter guess 2: 45
Wrong
```

```
Enter max # guesses: 15
Enter guess 1: 5
Wrong
Enter guess 2: 42
Right!
```

Code in bold, italics was to be filled in by students

```
void main() {
   int nGuesses;      // Maximum number of guesses
   int randValue;           // Actual random value
   int userGuess;           // Value guessed by user
   int i;                   // Loop index

   printf("Enter max # guesses: ");
   scanf("%d", &nGuesses);

   randValue = rand() % 100 + 1; // Value between 1 and 100

   // Loop that tracks current number of guesses
   for (i = 0; i < nGuesses; i++) {

      // Prompt for and read guess
      // MUST FILL IN BLANK IN PRINTF() WITH CORRECT EXPRESSION
      printf("Enter guess %d: ", i+1);
      scanf("%d", &userGuess);

      // Test to see if user correctly guessed value—if so, print
      //   "Right!" and then exit program
      if (userGuess == randValue) {
           printf("Right!\n");   // User correctly guessed value
           break;                // return also works
      }

      // Handle case where user guess is incorrect
      else
           printf("Wrong\n");    // User guess was incorrect
   }
}
```