Written Exam 3 Sample Problems – Answer Key

CSSE 221 – Fundamentals of Software Development Honors
Fall 2008-2009

1. [5 points] **C versus Java: biggest differences.** Sample problems:
   a. List three of the biggest differences between C and Java. Answer:
      - C does not have classes (it has only structures)
      - C does not do memory reclamation automatically (Java has a garbage collector)
      - In C, pointers are explicit (they are declared and dereferenced using the * notation)

   b. What happens when the following executes in C? In its equivalent in Java?

```
int x[3];
int y;
x[3] = 48;
```

   Answer: In C, the place in memory just beyond the array x gets the value 48. If that space is operating system space, the system gives an operating system error; if that space is within the program, whatever variable is stored at that place is overwritten.

   In Java, an IndexOutOfBoundsException is generated.

   c. What happens when the following executes in C? Can its equivalent happen in Java?

```
int* x;
int y;
x = 10;
y = *x;
```

   Answer: In C, the place in memory position 10 is accessed (in preparation for setting y to the contents at that position). Since that space is probably reserved for the operating system, the system gives an operating system error.

   Its equivalent cannot happen in Java because you cannot access a place in memory via its address.

2. [5 points] **Pointers: basics.** Sample problems:
   a. Declare a pointer to a float. Answer: float* p;
   b. Suppose x is a pointer to a float. Print the value of x, followed by the value of the float that x refers to. Answer: printf("%o %f", x, *x);
   c. Suppose that y is a char. Print the address of y. Answer: printf("%o", &y);
3. [5 points] **Obtaining information from a function:** returning values from the function and returning information through pointer arguments. Sample problems:
   a. Explain, by giving an example, how functions "normally" return values. Give both the function prototype and an example of the function call. Answer:

```
double foo(); // prototype
double x;
x = foo();     // call to function
```

   b. Repeat the previous problem, this time using a *void* function. Answer:

```
void foo(double* p);        // prototype
double x;
foo(&x);                    // call to function
```

4. [10 points] **Arrays versus pointers, and pointer arithmetic.** Sample problems:
   a. Consider a function whose specification and prototype is:

```
// Returns the sum of the numbers in the given array.
double foo(double* arr, int arrayLength);
```

   - Implement the function using array (bracket) notation. Answer:

```
double foo(double* arr, int arrayLength) {
    int k;
    double sum;

    sum = 0.0;
    for (k = 0; k < arrayLength; ++k) {
        sum += arr[k];
    }

    return sum;
}
```

- Implement the function using pointer arithmetic. Answer (with changes from the previous part in red):

```
double foo(double* arr, int arrayLength) {
    double* k;
    double sum;

    sum = 0.0;
    for (k = arr; k < arr + arrayLength; ++k) {
        sum += *k;
    }

    return sum;
}
```

5. [10 points] **Dynamic allocation of arrays.** Sample problems:

   a. Is the following code legal in C? Why or why not? Is its equivalent legal in Java?

```
void foo(int n) {
    double x[n];
    ...
}
```

   Answer: It is not legal in C, because static arrays must have their size specified at compile-time.

   b. Write a statement that declares an array of floats and allocates space for 300 floats in the array. Answer: float* arr = (float*) malloc(300 * sizeof(float);

   c. Write a statement that declares a C-style string and allocates space for it to contain up to 300 characters.

   Answer: char* arr = (char*) malloc(301 * sizeof(char);

   Note the space for the \0 that terminates the string.

   d. Write the statement that deallocates (i.e. makes available again) the space allocated in the previous problem. Answer: free(arr);

   e. Explain the difference between how space is deallocated (i.e. made available again) in C and in Java. Answer: In Java, when an object reaches the situation where nothing in the program can refer to that object any more, then the garbage collector reclaims the space allocated to that object. In C, the programmer is required to free that space herself when the space is no longer needed.

6. [10 points] **Structures.** Sample problems:

   a. What is the most obvious difference between structures in C and classes in Java?
   Answer: structures do not have functions (methods) associated with them.

   b. Write a statement that declares a structure type Car that contains the Car's model (at most 20 characters), year, and gasoline tank capacity (to the nearest tenth of a gallon). Answer:

```
typedef struct {
   char model[20];
   int year;
   float tankCapacity;
} Car;
```

   c. Write statements that declares a Car variable and gives values to its fields.

```
Car c;
strcpy(c.model, "ford");
c.year = 1921;
c.tankCapacity = 4.5;
```

   d. Write statements that declares a pointer to a Car variable, allocates space for the variable to hold a Car, and gives values to the Car's fields.

```
Car* c;
c = (Car*) malloc(1 * sizeof(Car));
strcpy(c->model, "ford");
c->year = 1921;
c->tankCapacity = 4.5;
```

   e. How would you change the above subproblems if you wanted to allow the Car's model to be a string of unlimited length?

```
typedef struct {
   char* model;
   int year;
   float tankCapacity;
} Car;
c.model = (char*) malloc(DESIRED_SIZE * sizeof(char));
c->model = (char*) malloc(DESIRED_SIZE * sizeof(char));
```

f. Consider the following code:

```
typedef struct {
    int x;
    int y;
    int z;
} Point3D;

Point3D p1;
Point3D p2;

p1.x = 10;
p1.y = 20;
p1.z = 30;

p2 = p1;
```

Is p2 a copy of p1, or do p2 and p1 refer to the same place in memory? What about the equivalent situation in Java? Answer: p2 is a copy of p1; its 3 fields are copies of the 3 fields of p1 and do NOT refer to the same places in memory as p1's fields. This is different than in Java, where p1 and p2 would both refer to the same place in memory.

g. Continuing the previous problem, consider the following code:

```
Point3D p1;

p1.x = 10;
p1.y = 20;
p1.z = 30;

foo(p1);
```

When execution returns after the call to foo, is p1.x guaranteed to be 10, or might foo have changed the value of p1.x? What about the equivalent situation in Java?

Answer: p1.x is guaranteed to be 10 – the call foo(p1) made a copy of p1 to send to foo. This is not the case in Java, where p1 is a reference (pointer) to a place in memory (and hence can have its fields changed).

h. Consider the code:

```
typedef struct {
    float*   w;
    float    x;
    char*    y;
    char[10] z;
} Trio;

Trio t;
```

How much storage is allocated by the declaration of t? That is, how many floats? Chars? Etc?

Answer: The declaration allocates:

- 1 float
- 10 characters
- 2 pointers

7. [5 points] **Strings.** Sample problems:

   a. What is the terminating character of a C-style string?

   Answer: \0 (backslash zero)

   b. Declare a C-style string and give it value *java*. Answer:

```
char s[] = "java";
```

or

```
char* s;
s = (char*) malloc(5 * sizeof(char));
strcpy(s, "java");
```

or

```
char s[5];  s[0] = 'j';  s[1] = 'a';  s[2] = 'v';  s[3] = 'a';  s[4] = '\0';
```

   c. What is the minimum number of characters that you must allocate to solve the previous problem? Answer: 5 (NOT 4 – you must leave space for the terminating backslash zero).

d. Consider the following code:

```
char* x = "jumbled";
char* y = x;
stringSort(y);  // Sorts y, so that y becomes "bdejlmu"
```

Explain why the above code changes x. Then fix the code so that x does not change (but y still becomes the sorted version of x). Answer:

The second line makes y point to the same place in memory that x points to. Hence, sorting y also sorts x.

To fix the problem:

```
char* x = "jumbled";
char* y = (char*) malloc((strlen(x) + 1) * sizeof(char));
strcpy(y, x);
stringSort(y);
```