CMSC 106 Quiz 7 Worksheet

The next quiz for the course will be on Wed, Nov 30. The following list provides more information about the quiz:

- The quiz will be a written quiz (no computer).
- Closed book, closed notes quiz.
- Answers must be neat and legible.
- Quiz instructions can be found at http://www.cs.umd.edu/~nelson/classes/utilities/examRules.html

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TA or instructor during office hours. If is recommended that you try these exercises on paper first (without using the computer).

Exercises

1. Define a structure (using typedef) called **Name** that has the following fields:

   first_name – maximum length 80
   last_name – maximum length 80

2. Define a **compare_name** function that takes two Name structures and returns -1 if first parameter precedes the second, 0 if they are equal and 1 otherwise. The function should compare **last_name** first.

3. Define a structure (using typedef) called **Customer** that has the following fields:

   full_name – of type Name (defined above)
   age → integer
   phone → string (maximum size 12)

4. Define a function **init_customer** that initializes and returns a **Customer** structure. The parameters to the function will be full_name, age and phone.

5. Define a function called **print_customer** that prints the contents of a **Customer** structure.

6. Define a function called **equals** that compares two **Customer** structures. Two customers are considered equal if they have the same name. The function takes two **Customer** pointers as parameters and returns 1 if the customers are equal and 0 otherwise.

7. Define a function called **find_customer** that takes an array of **Customer** structures as parameter. The function will return 1 if a customer with a particular name is found in the array and 0 otherwise.

8. The following structure definitions will be used for the questions below:

```
#define MAX_LENGTH 80

typedef struct song {
   char title[MAX_LENGTH + 1];
   int duration;
} Song;
```

   a. Implement a function called **init_song** that has the prototype below. The function will return a **Song** structure initialized with the parameter values. To copy the title use strcpy.

```
Song init_song(char *title, int duration)
```

   b. Implement a function called **print_song** that has the prototype below. The function will print the title followed by the duration. For example, for a song whose title is "Hello" and has a duration value of 4, the function will print "Hello 4". Notice the function has as parameter a pointer.

```
void print_song(Song *ptr)
```

   c. Implement a function called **find_song** that has the prototype below. The function will return 1 if there is a song in the array with the specified title and 0 otherwise. You can assume title songs are unique.

```
int find_song(Song array[], int array_size, char *title)
```

9. The following structure definitions will be used for the questions below:

```
#define MAX_LENGTH 80

typedef struct computer {
   char model[MAX_LENGTH + 1];
   double cost;
} Computer;
```

a. Implement a function called **init_computer** that has the prototype below. The function will return a **Computer** structure initialized with the parameter values. To copy the model use strcpy.

```
Computer init_computer(const char *model, double cost)
```

b. Implement a function called **print_computer** that has the prototype below. The function will print the model and cost for the specified computer separated by a single space (e.g., delly 400.000000). If the pointer parameter is NULL the function will print the message "No computer provided".

```
void print_computer(Computer *ptr)
```

c. Implement a function called **get_computer_count** that has the prototype below. The function will return the number of computers that have a cost that is less than the cost parameter.

```
int get_computer_count(Computer array[], int array_size, double cost)
```