CMSC 216 Quiz 4 Worksheet

The next quiz for the course will be on Wed, Mar 23. The following list provides additional information about the quiz:

- The quiz will be a written quiz (no computer).
- The quiz will be in lab session.
- 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
- Make sure you know your section number and your TA's name.
- You must take your quiz in your assigned lab/discussion section and not show up to a random discussion section. We will not grade quizzes taken in the incorrect section.
- **Regarding Piazza -** Feel free to post questions in Piazza regarding the worksheet and possible solutions to problems, but for coding questions please do not post code. You can post suggestions on how to solve coding problems, but your classmates will benefit more if they themselves actually solve the problems. Pretend you are a TA while addressing or providing help in Piazza ☺

At the end we have provided an example of a memory map so you know exactly what we are expecting while drawing maps. Take a look at the example before drawing any maps.

IMPORTANT: Notice how we represent an array in the map (we just add the name next to the first element followed by the elements). In class (for Nelson's sections) we were using a cell pointing to the first element to show what the compiler does when passing the array to a function, but when drawing the memory map we do not want to define such a cell.

Exercises

1. What situation has occurred after the last assignment of the following code?

```
int *p = malloc(sizeof(int));
*p = 400;
p = NULL;
```

2. How are the malloc and calloc functions different?

3. Name one advantage of initializing pointer variables to NULL?

4. What is the problem in the following code?

```
double scores[100];
double *p = scores;
scores[0] = 13.2;
free(p);
```

5. Implement the **append** function that has the prototype below. The function returns a string that represents the concatenation of all the strings present in an array of strings. For this problem, you can assume the end of the parameter array is marked by NULL. You need to allocate memory for the resulting string. You may not modify the array parameter.

```
char* append(char *data[]);
```

6. Write a function named **longest** that has the prototype below. The function returns a **copy** of the longest string in an array of strings. The end of the array is marked by an entry with a value of NULL. You can assume the **data** array will have at least one string. The function must free all the memory associated with the **data** array.

```
char* longest(char **data);
```

7. The following structures will be used for the questions that follow:

```
#define MAX_LEN 80

typedef struct {
    char *title;
    int duration;
} Song;

typedef struct {
    Song* all_songs;  /* array of songs */
    int number_of_songs;
    char album_name[MAX_LEN + 1];
} Album;
```

   a. Given two Song structures s1 and s2, are there any problems with assigning one structure to another? Is deep copying taking place during the assignment of these two structures?

   b. Define a function that initializes a Song using a duration and a string provided. The function will make a copy of the string parameter.

   c. Define a function that will initialize an Album structure based on an array of Songs provided as parameter. The function will create a dynamically-allocated array of Songs and will initialize each entry with a copy of the corresponding song in the parameter array.

8. The following structures will be used for the questions below.

```
typedef struct car {
   char *make;
   int tag;
} Car;

typedef struct {
   Car **lanes;
   int max_lanes;
} Beltway;
```

   a. Define a create_beltway function that has the prototype below. The function will define an array of length max_lanes of Car *, and initialize them to null.

```
void create_beltway(Beltway **beltway, int max_lanes)
```

   b. Define a function **add_car** that will add a Car object to the lane with the smallest number of cars.

   c. Define a function **increase_beltway** that will add a number of empty lanes to the beltway.

   d. Define a function **duplicate** that will create a full duplicate of a beltway.

   e. Define a function **destroy** that will give back any dynamic memory associated with a beltway structure to the memory allocator.

Sample Memory Map

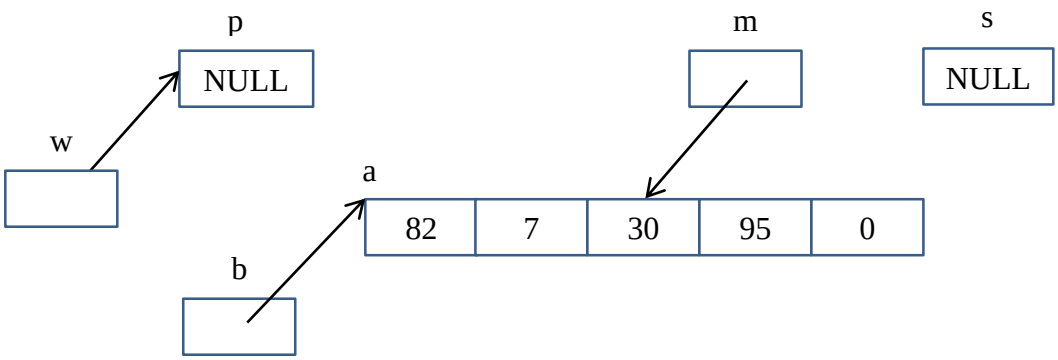
We are providing this example so you know what we are expecting for memory maps.

Example

Draw a memory map for the following program up to the point indicated by the comment **/*HERE */**.

```
#include <stdio.h>

#define MAX_LEN 5

void process(int *b, int *s, int **w) {
    b[0] = 82;
    s[1] = 95;
    s = NULL;
    *w = NULL;
    /* HERE */
}

int main() {
   int a[MAX_LEN] = {10, 7, 30, 40};
   int *p = a;
   int *m = a + 2;

   process(p, m, &p);

   return 0;
}
```

Answer:



Note: You can also replace NULL with the ground symbol as done in lecture. For example, s above could be represented as: