

CMSC 216 Quiz 2 Worksheet

The next quiz for the course will be on Wed, Feb 27. 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>
- **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.

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.

Exercises

1. What takes place when an array name is assigned to a pointer variable?
2. What is the output of the following code fragment? Assume an integer occupies 4 bytes.

```
if (sizeof(char *) == sizeof(int *)) {  
    printf("One\n");  
} else {  
    printf("Two\n");  
}
```
3. Why we cannot dereference a void pointer?
4. The address stored by a pointer variable represents the address of the first byte of the entity the pointer refers to. **True** or **False**.
5. When do you want to use the const modifier?
6. What requires more effort in C: passing an array of two elements or passing an array of a million elements?
7. What is the difference between a string and a character array?
8. How do you define a string variable in C?
9. What is the difference between NULL and the '\0' character?
10. Define a function called **remove_char** that has as parameters a string (named **src**), a character (named **target**) and a string output parameter (named **result**). The function will initialize the **result** parameter with a string where the first occurrence of the **target** character, if present, has been removed. You can assume the **result** parameter is large enough to fit the answer.
11. Define a function called **find_char** that has as parameters a string (named **src**) and a character (named **target**). The function will return a pointer to the first instance of **target** in the **src** string, and NULL otherwise. Provide a recursive implementation (no auxiliary function is allowed).
12. Rewrite the previous **find_char** function so the pointer (or NULL) that the function returns is returned via an output parameter that is a char ** pointer. This output parameter will be used to initialize a pointer variable that exist at the point where the function is called.
13. Write a function called **repeat_n** that takes as parameters a string (named **src**), an integer (named **n**), and an output string parameter (named **result**). The function will initialize the output parameter with **n** instances of the **src** string. If the length of the **src** string is greater than 20 or it is equal to the word "password", no processing will take place. Feel free to use the strcpy, strlen and strcmp string library functions. You can assume the **result** parameter is large enough to fit the answer.
14. Finding Pepsi on campus is really easy; finding coke is very difficult. ☺

15. Draw a memory map for the following program at the point in the program execution indicated by the comment **/*HERE */**. In addition, provide the output generated by the program.

```
#include <stdio.h>

#define MAX_LEN 5

static void task(int *b, int range) {
    b[range - 1] = 200;
    range = 0;
    b = NULL;
    /* HERE */
}

int main() {
    int a[] = {2, 4, 6};
    int len = 3, i;

    task(a, len);
    printf("len %d\n", len);
    for (i = 0; i < len; i++) {
        printf("%d\n", a[i]);
    }

    return 0;
}
```

16. Draw a memory map for the following program at the point in the program execution indicated by the comment **/*HERE */**.

```
#include <stdio.h>

#define MAX 4

static void work(int *b, int delta) {
    int i = 0;

    for (i = 0; i < delta; i++) {
        b[i] += 1;
    }
    delta = 0;
    *b = 999;
    b = NULL;

    /* HERE */
}

int main() {
    int x = 50, *p = &x, eval = 2, a[MAX] = {7, 11, 3};
    float y = 30, *m = &y, *t = m;

    if (sizeof(p) == sizeof(m)) {
        x += 100;
    } else {
        x += 200;
    }
    *t += 4;
    *m += 5;

    work(a, eval);

    return 0;
}
```

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 at the point in the program execution indicated by the comment **/*HERE */**.

```
#include <stdio.h>

#define MAX_LEN 5

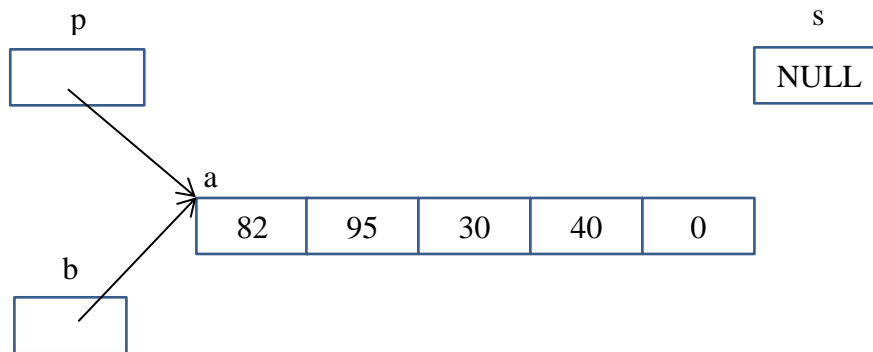
void process(int *b, int *s) {
    b[0] = 82;
    s[1] = 95;
    s = NULL;
    /* HERE */
}

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

    process(a, 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:

