**CMSC 216 Quiz 2 Worksheet**

The next quiz for the course will be on Thu, Jun 13. 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](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 is a NULL pointer?
2. Which of the following pointer variables occupies the largest number of bytes?
   
   ```
   int *x;
   float *y;
   double *m;
   ```
3. Why do we need to specify the type of a pointer variable?
4. How many memory locations can a pointer variable point at, at any given time?
5. Why will a segmentation fault occur when we dereference NULL? For example,
   
   ```
   int *ptr = NULL;
   printf("%d\n", *ptr);
   ```
6. How are pointer arguments to functions passed in C? By value? By reference?
7. What is the output of the following program? Would it be possible to get a segmentation fault?
   
   ```
   #include <stdio.h>

   int main() {
       int *ptr;
       *ptr = 400;
       printf("%d\n", *ptr);
       return 0;
   }
   ```
8. Write a code fragment that shows that NULL is considered false in C.
9. What takes place when an array name is assigned to a pointer variable?
10. 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");
    }
    ```
11. The following program compiles.

```c
#include <stdio.h>

int main() {
    int x;
    int *p = &x;
    printf("%d", *p);
    return 0;
}
```

What would happen when we execute the program?

a. A segmentation fault will always occur.
b. The value 0 will be printed.
c. A garbage/trash value will be printed, but no segmentation fault will take place.
d. Sometimes a garbage/trash value will be printed and sometimes a segmentation fault will take place.
e. None of the above.

12. The address stored by a pointer variable represents the address of the first byte of the entity the pointer refers to. **True** or **False**.

13. When do you want to use the const modifier?

14. What requires more effort in C: passing an array of two elements or passing an array of a million elements?

15. Finding Pepsi on campus is really easy; finding coke is very difficult. 😊

16. 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.

```c
#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;
}
```
17. Draw a memory map for the following program at the point in the program execution indicated by the comment /*HERE*/.

```c
#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*/.

```c
#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:

```
   p
    |
   ---
    |
   a
   |
 b   82 95 30 40 0
    |
    s
    |
   NULL
```

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

```
s
    |
    ---
    |
    ---
   __
```