CMSC 216 Quiz 2 Worksheet

The next quiz for the course will be on Wed, Feb 12. 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.

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 this exercises on paper first (without using the computer).

Exercises

1. You need to be familiar with the style guide provided at:

   http://www.cs.umd.edu/class/spring2014/cmsc216/resources/coding-style.html

2. When should a function be defined as static?

3. Name one type of variable that relies on static storage.

4. What is the output of the following code fragment?

```
int value = 7;
printf("%ld\n", sizeof(++value));
printf("%d\n", value);
```

5. Which of the following pointer variables uses the largest number of bytes?

```
int *p;
char *q;
```

6. What will happen when the following code is executed? Explain briefly.

```
int *p = NULL;
*p = 20;
```

7. What will happen when the following code is executed? Explain briefly.

```
int *p;
*p = 20;
```

8. What will happen when the following code is executed? Notice **a** has not been initialized Explain briefly.

```
int a;
int *a_ptr = &a;

printf("%p\n", a_ptr);
```

9. What is the difference between a void pointer variable and a non-void pointer variable (e.g., an integer pointer variable)?

10. Write a memory map for the program below. In cases where you are asked to print an address, write **NULL** or **MEMORY_ADDRESS** (for any other value). To help you understand pointers better, you may assume some memory addresses while drawing the memory map (as we did in lecture).

```
#include <stdio.h>

void process(float *passed_card);
void analyze(float **evaluated_card);

void process(float *passed_card) {
   *passed_card += 200;
   printf("In process_one %.2f\n", *passed_card);
   passed_card = NULL;
}

void analyze(float **evaluated_card) {
   if (**evaluated_card >= 600) {
      *evaluated_card = NULL;
   } else {
      evaluated_card = NULL;
   }
}

int main() {
   float bank_account = 500.00;
   float *card_one, *card_two, *card_three = &bank_account;

   card_one = card_two = &bank_account;
   printf("V1: %.2f\n", bank_account);
   printf("V2: %.2f\n", *card_one);
   printf("V3: %.2f\n", *card_two);
   printf("V4: %p\n", (void *)card_two);
   printf("V5: %p\n", (void *)&bank_account);

   *card_two += 20.0;
   printf("V6: %.2f\n", *card_two);
   card_two = NULL;
   printf("V7: %.2f\n", *card_one);

   process(card_one);
   printf("V8: %p\n", (void *)card_one);
   printf("V9: %.2f\n", bank_account);

   analyze(&card_three);
   printf("V10: %p\n", (void *)card_three);
   printf("V11: %.2f\n", *card_one);

   return 0;
}
```

11. Provide the output of the previous program.