CSSE 132 – Introduction to Systems Programming
Rose-Hulman Institute of Technology

Homework 5

Name (Print):_____ RHIT Username:_____

Put your answers in the spaces reserved below the questions. **Do not** write your answers on **separate sheets of paper**.

Note: The code samples below frequently use the function `memcpy`. Please use command `man memcpy` to check the manual page for this function.

1. (5 points) Consider the code below:

```c
#include <stdlib.h>
#include <stdio.h>

void main(int argc, char** argv) {
  int a = 5;
  printf("a is %d\n", a);
  free(&a);
}
```

   This code crashes when it is run. Why does this happen?

2. (8 points) Below is a simple test program for testing a copy function.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* copier(char* src) {
  char* tmp = malloc(strlen(src) + 1);
  memcpy(tmp, src, strlen(src) + 1);
  return tmp;
}

void main(int argc, char** argv) {
  char* copy = copier(argv[1]);
  printf("input: %s\ncopy: %s\n", argv[1], copy);
}
```

The test program above produces the correct output and seems to runs without issues. However, when the copy function is deployed to a production server the server crashes after running for a few hours.

(a) The pointer `tmp` points to memory memory in which segment of memory: *stack*, *heap*, or *data*?

(b) Why do you think the server crashes? Why did the test not catch this?

(c) What could you do to fix the problem?

3. (7 points) You decide to write a simpler copy program, but it also has problems! This code seems to work most of the time, but on some input strings it has strange output—for example, printing only part of the string followed by strange unprintable characters.

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char** argv) {
  char tmp[128];
  if (argc < 2) return 0;

  memcpy(tmp, argv[1], 128);
  printf("input: %s\ncopy: %s\n", argv[1], tmp);

  return 0;
}
```

(a) The pointer `tmp` points to memory memory in which segment of memory: *stack*, *heap*, or *data*?

(b) What types of input strings may cause the program to have strange output, and why?

(c) How can we avoid the strange output for these cases?

4. (4 points) Consider the code below:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* printOnce() {
  char* c = malloc(10);
  memcpy(c, "Amadeus!!", 10);
  printf("%s\n", c);
  free(c);
  return c;
}

int main(int argc, char** argv) {
  char* c;
  c = printOnce();
  printf("%s\n", c);  // print again
  return 0;
}
```

   Should we expect this code to consistently print the string the second time? Why or why not?

5. (24 points) On your Linux, pull changes to your Git repository. Complete the C functions in the `problems.c` files in the `homework5` subdirectory. The comments tell you what needs to be completed. Use `make test` and `./test` to compile and test your code.

   When finished, submit your `problems.c` to **Gradescope**.