CSSE 132 – Introduction to Computer Systems
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Homework 5

Write your answers on a *separate sheet of paper.* Show your work, and when writing code, make sure you document it well.

1. (10 points) Read the following code and answer the questions.

```
void mystery0(int a, int b, int c)
{
  a = b + c;
}

void mystery1(int* a, int* b, int* c)
{
  *a = *b + *c;
}

int mystery2(int* d, int e, int f)
{
  int i = 0;
  int j = 0;
  for (i = 0; i < f; i = i + e) {
    j = j + d[i];
  }
  return j;
}
```

(a) What is the value of x after the following code runs?

```
int x = 0;
int y = 1;
int z = 20;
mystery0(x, y, z);
```

(b) What is the value of x after the following code runs?

```
int x = 0;
int y = 1;
int z = 20;
mystery1(&x, &y, &z);
```

(c) Explain the difference between mystery0 and mystery1.

(d) If `arr` is an array containing the following values: `{1, 2, 3, 4, 5, 6, 7, 8, 9}`, what does `mystery2(arr, 2, 9)` return?

(e) What would be a more descriptive name for the function `mystery2`?

2. (5 points) Consider the code below:

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

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

  return 0;
}
```

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

3. (5 points) Consider the code below:

```
int* add(int* a, int* b)
{
  int c = *a + *b;
  return &c;
}

int main(int argc, char** argv)
{
  int a = 5;
  int b = 6;
  int *c;
  c = add(&a, &b);
  return *c;
}
```

This code occasionally returns the correct answer of 11, but sometimes does not. Explain why this happens.

4. (10 points) Below is a simple test program for testing a copy function.

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

char* copier(char* src)
{
  char* tmp = malloc(strlen(src)+1);
  strncpy(tmp,src,strlen(src)+1);

  return tmp;
}

int main(int argc, char** argv)
{
  char* copy = copier(argv[1]);
  printf("input: %s\ncopy: %s\n",argv[1],copy);
  return 0;
}
```

The test program above produces the correct output and seems to runs without issues. However, when the copy function is used in a production server everything works fine at first, but the server crashes after running for a few hours.

(a) Where in memory is the tmp string stored?

(b) Why does the test appear correct, but the server crash?

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

5. (10 points) You decide to write a simpler copy program, but it also has problems! This code seems to work most of the time, but can still fail.

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

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

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

  return 0;
}
```

(a) Where in memory is the tmp string stored?

(b) What is printed after the code runs with input “AAAA”?

(c) Are there strings that can cause problems?

(d) Find what is missing and explain why it should be there.

6. (5 points) Consider the code below:

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3
 4 struct db_entry {
 5   char* name;
 6   char* value;
 7 };
 8
 9 void dbe_print(struct db_entry* entry)
10 {
11   fprintf(stdout, "%s =>%s\n", entry->name, entry->value);
12 }
13
14 int main(int argc, char** argv)
15 {
16   struct db_entry* e;
17   e->name = "hocus pocus";
18   e->value = "focus";
19   dbe_print(e);
20   return 0;
21 }
```

This code compiles, but crashes when it is run. Suspecting a problem with `dbe_print`, you comment out the call that prints the entry, however the program still crashes! You suspect the crash is caused by a memory issue.

(a) What is the reason for the crash?

(b) Without changing the above lines, insert code to correct the crash and resolve the memory issue. Indicate the line numbers where your code will be inserted.

7. (5 points) Consider the code below:

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

int main(int argc, char** argv)
{
  FILE * f;
  int size = 0;
  char* filename = argv[1];

  /* Your code goes here */

  printf("File size is %d\n", size);
  return 0;
}
```

Using standard C buffered I/O, add code to open `filename`, get the file size so that it can be printed out, and close the file.

8. (5 points) Consider the code below:

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

char* printOnce()
{
  char * c = malloc(10);
  strncpy(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;
}
```

This code prints the string twice most of the time. However, sometimes the program crashes. Why does this happen?