lab 05 Pointers

Instructions: It is ***vital*** that you understand pointers and, consequently, the different memory spaces: static, stack, and heap. Please answer the following questions:

Note: When a question asks for the *value* of a variable, if it is a known number write the number. If it is a memory address write what the memory address is pointing to (Ex: The variable x holds the memory address of the variable y.) If it cannot be determined, write undefined.

1. (15 pts) Consider the following code:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    int x = 32;
    int *y = &x;
    int z = *y;
    printf("z = %d\n", z);
    return 0;
}
```

   (a) What is the output?
   (b) What is the *value* of y?
   (c) In what memory (static, stack, or heap) do the variable x, y, z exist during runtime?

2. (15 pts) Consider the following code:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    int *x = (int *)200;
    long z = (long)x;
    printf("z = %d\n", z);
    return 0;
}
```

   (a) What is the output?
   (b) In what memory (static, stack, or heap) do the variable x, z exist during runtime?
   (c) Why is this code bad practice (even though it compiles/runs without a seg fault)?

3. (15 pts) Consider the following code:

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4      int *x = new int[100];
5      x[0] = 500;
6      int z = x[25];
7      printf("z = %d\n", z);
8      return 0;
9  }
```

(a) What is the output? (Careful, a compile/run will not give you the correct answer.)

(b) In what memory (static, stack, or heap) do the variable x, z exist during runtime?

(c) What is stored on the heap?

(d) What is the memory issue?

4. (15 pts) Consider the following code:

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4      int *x = new int[100];
5      int *y = x+10;
6
7      for (int i = 0; i < 100; i++) {
8          x[i] = i;
9      }
10
11     printf("y[10] = %d\n", y[10]);
12     return 0;
13 }
```

(a) Why is the output?:
    $y[10] = 20$

(b) Write a line of code that would free up memory using only x.

(c) Write a line of code that would free up memory using only y.

5. (15 pts) Consider the following code:

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    int *x = new int[100];
    int *y = new int[100];
    int **z = NULL;

    for (int i = 0; i < 100; i++) {
        x[i] = i;
        y[i] = 100-i;
    }

    z = &x;
    printf("(*z)[10] = %d\n", (*z)[10]);
    z = &y;
    printf("(*z)[10] = %d\n", (*z)[10]);
    return 0;
}
```

(a) What is the *value* of x and y?

(b) What is the *value* z on lines 14 and 16?

(c) What does the code fragment "(*z)[10]" mean? (Describe what the code must do the evaluate that code fragment.)

(d) Why are there two different outputs for "(*z)[10]"? Here is the output of the program:
(*z)[10] = 10
(*z)[10] = 90

(e) Write the commands to free memory.

6. (25 pts) Consider the following code:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  class Student {
5      public:
6          int mId;
7          double mGPA;
8  };
9
10 int main(int argc, char *argv[]) {
11     Student *students = new Student[100];
12     Student ** studentsPtr = new Student*[100];
13
14     srand(100); // Seed random number generator
15     for (int i = 0; i < 100; i++) {
16         students[i].mId = i+1;
17
18         // Generate a "random" GPA from 0.0-4.0
19         students[i].mGPA = 4 * (((double)rand())/RAND_MAX);
20
21         studentsPtr[i] = students+i;
22     }
23
24     // This is Bubble Sort:
25     for (int i = 0; i < 100; i++) {
26         for (int j = 1; j < 100; j++) {
27             // Based on GPA
28             if (studentsPtr[j-1]->mGPA > studentsPtr[j]->mGPA) {
29                 Student *temp = studentsPtr[j];
30                 studentsPtr[j] = studentsPtr[j-1];
31                 studentsPtr[j-1] = temp;
32             }
33         }
34     }
35
36     for (int i = 0; i < 100; i++) {
37         printf("%f\n", studentsPtr[i]->mGPA);
38     }
39
40     return 0;
41 }
```

(a) What is the *value* of students?

(b) What is the *value* of studentsPtr?

(c) Which of the above is being sorted?

(d) Write the code to sort students without nesting for loopss. (Hint: use studentsPtr.)

(e) Write the commands to free memory.

How to turn in:
Turn in via GitHub. Ensure the file(s) are in your directory and then:

- $ git add <files>

- $ git commit

- $ git push

Due Date: September 18, 2019 2359

Teamwork: No teamwork, your work must be your own.