CSCI 1730 Breakout Lab 06

Pointers, Strings, Arrays, and Memory Maps

Learning Outcomes

- Demonstrate knowledge of pointers and arrays by tracing through the output of source code containing pointers to 1D arrays.
- Demonstrate knowledge of simple data structures (strings and arrays) by writing out their memory maps and tracing through the output of source code.
- Use memory maps to understand the memory model of a C process.

Problem / Exercise

Pointers are an integral part of the C and C++ programming languages. While they may be the cause of many headaches, they allow for a great deal of control and flexibility. Because they are low-level constructs, they require a solid understanding of the memory model behind it all. Not every language has pointers, but what you learn about memory management will apply to most programming languages. This lab is structured differently from previous labs. You'll be asked to observe some code and answer questions about that code or its output. Also, you will fill out memory maps on a separate piece of paper.

1 Pointers Activity

Read and understand the following C program (assume its was compiled on odin using the required compiler for this class), and on a piece of paper, write out memory maps for each variable as the code is executed on odin line-by-line in the main function and any other function calls. The memory map should include each variable's name, its address (you may assume the first variable's address is 100, the second variable's address is 200, the third variable's address is 300, etc.), and its value. If the variable represents an array, assume a reasonable starting address as aforementioned, then list the address of each element in the array correctly. The memory map should change as new variables are declared, initialized, and modified throughout the line-by-line execution of the main function and any other function calls.

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

void f(int x, int * y);

int main(void){
  char s1[] = "zoey";
  char s2[] = "zoey";
  char * s3 = s1;
  char * s4 = s2;
  printf("1.  sizeof(s1) = %lu\n",  sizeof(s1));
  if(s1 == s2) printf("2.  s1 == s2 is true\n");
  else printf("2.  s1 == s2 is false\n");
  if(strcmp(s3, s4) == 0) printf("3.  (strcmp(s3, s4) == 0) is true\n");
  else printf("3.  (strcmp(s3, s4) == 0) is false\n");

  double d[5] = {7.5, 2.8, 9.5, 88.7, 1.7};
  int v[4];
  for(size_t i = 0; i < 4; i++){
    *(v + i) = (int) ((unsigned long) &d[i+1] - (unsigned long) &d[i]);
  }
```

```
  printf("4.  The values stored in v are ");
  for(int i = 0; i < 4; i++){
    printf("%d ", v[i]);
  }
  printf("\n");

  char * p1 = &s4[0];
  *p1 = 'j';
  *(&s4[3]) = 's';
  char * s5 = "zoey\0 is awesome";
  printf("5.  s4 = %s, s5 = %s\n", s4, s5);

  int q[5];
  int * p2 = q;
  for(int i = 1; i < 6; i++) *(p2 + i - 1) = i * 5;
  f(*(q + 1), (q + 2));
  printf("6.  What is the value of (q[1] + q[2])?\n");
  printf("7.  p2 = %ld, q = %ld\n", (unsigned long) p2, (unsigned long) q);
  printf("8.  p2 = %p, q = %p\n", (void *) p2, (void *) q);
  return 0;
}

void f(int x, int * y){
  x = -2;
  *y = -3;
}
```

An output for the C program above executed on odin.

```
1.  sizeof(s1) = 5
2.  s1 == s2 is false
3.  (strcmp(s3, s4) == 0) is true
4.  The values stored in v are 8 8 8 8
5.  s4 = joes, s5 = zoey
6.  What is the value of (q[1] + q[2])?
7.  p2 = 140736793682416, q = 140736793682416
8.  p2 = 0x7fffd69821f0, q = 0x7fffd69821f0
```

Create a file called lab06Answers.txt on odin, write your first and last name (as they appear on eLC) on the first line of this file, write your 811 or 810 number on the second line of this file, and on subsequent lines in this file, answer the following seven numbered questions based on the C program and an output for it above. Many of the numbered questions below have multiple parts, and students must answer all parts with explanations to receive full credit for that numbered question.

1. In the output for 1 above, why is sizeof(s1) = 5 even though s1 contains only four letters? What character is being stored in the fifth byte of s1?

2. In the output for 2 above, why does s1 == s2 result in false when both s1 and s2 contain the same character sequence? What values are being compared in s1 == s2 to yield a result of false?

3. In the output for 3 above, why does (strcmp(s3, s4) == 0) result in true?

4. In the output for 4 above, why are all of the values stored in v equal to 8? Explain your answer in terms of how the memory addresses are being stored for d. Assume the starting address of d is 500, then what are the memory addresses for d[0], d[1], d[2], d[3], and d[4]?

5. Why is s4 = joes in the output for 5 above instead of s4 = zoey? Why did the output for 5 above print s5 = zoey instead of s5 = zoey is awesome? Explain your answers to these questions.

6. As asked in the output for 6 above, what is the value of (q[1] + q[2])? Is it correct syntax to use *(q[1] + q[2]) to compute the value of (q[1] + q[2])? Explain why this syntax is correct or why it isn't correct.

7. Consider the output for 7 and 8 above. Why do these memory addresses typically change each time this program is run on odin? Which number system: binary, octal, base-10, or hexadecimal, is being used to display the memory addresses shown in 7 above? Which number system: binary, octal, base-10, or hexadecimal, is being used to display the memory addresses shown in 8 above?

2 Submission

Submit your file, lab06Answers.txt, before the due date and due time stated on eLC. Submit only this file on eLC for this assignment. No README.txt file is required for this lab assignment.

3 Breakout Lab Attendance: 3 points

Breakout lab attendance is required for this assignment, and it is worth three points. Students must physically attend the entire period of the breakout lab section that they registered for during the week of this assignment to earn these three points for attendance. If you complete the assignment before the end of your breakout lab period during the week of this assignment, then you are still required to attend the entire breakout lab period to earn attendance points, and you may use this time for independent study for the next exam in this course.

4 Grading: 7 points

Students are expected to do all activities, follow all instructions, answer all questions, and use memory maps for this lab. Furthermore, neatness and organization is expected (otherwise, we will deduct points). The following grading will be used.

All questions answered correctly with correct explanations	7 points
Not submitting to the correct Assignment on eLC	-1 point
Late penalty for submitting 0 hours to 24 hours late	-2 points
Not submitting this assignment before its late period expires	-7 points
Penalty for not following instructions (disorganization, etc.)	Penalty decided by grader