Report

Assignment 1

In assignment 1, I declared an integer variable and a pointer that points to its address (int *pointer = &num). I printed the address of the integer using both the variable's address-of operator (&num) and the pointer itself to demonstrate they are identical. Then, I used the pointer to dereference and change the value of the integer to 2, and verified the change by printing the value through the pointer (*pointer).

```
---------- ASSIGNMENT 1 ----------
The address of the integer using variable 0x7ffea67ae3dc
The address of the intgere using pointer 0x7ffea67ae3dc
The value of the freshly changed variable 2
```

Assignment 2

In assignment 2, I created an array of integers and made a pointer that points to the first element of the array (int *pointer = numbers). I traversed through the array and printed each element via a printf line of code in which I passed the element of the array using pointer notation (*(pointer + i)). Then, using the same syntax, I changed each element by adding 10 to it by traversing again. Next, I used pointer notation traversal again to show the updated results, and then I used the array name syntax (numbers[i]) to traverse once more.

```
---------- ASSIGNMENT 2 ----------
Element #1 is 1
Element #2 is 2
Element #3 is 3
Element #4 is 4
Element #5 is 5
Element #1 is 11
Element #2 is 12
Element #3 is 13
Element #4 is 14
Element #5 is 15
Element #1 is 11
Element #2 is 12
Element #3 is 13
Element #4 is 14
Element #5 is 15
```

Assignment 3

In assignment 3, I implemented a swap function to exchange the values of two integers. Instead of passing the variables directly, I passed their addresses (&num1, &num2) to the function. Inside the function, I used pointers to access the original memory locations, using a temporary integer to hold one value while I assigned the value of the second pointer to the first.

```
---------- ASSIGNMENT 3 ----------
num1 is 5 and num2 is 10
num1 is 10 and num2 is 5
```

Assignment 4

In assignment 4, I declared a pointer to a pointer (int **ptrToPtr). I pointed the first pointer to an integer and then pointed the second pointer to that first pointer. Using double pointer (**ptrToPtr), I reached the value of the integer

```
---------- ASSIGNMENT 4 ----------
The value of the integer using the pointer 5
The value of the integer using the pointer of the pointer 5
```

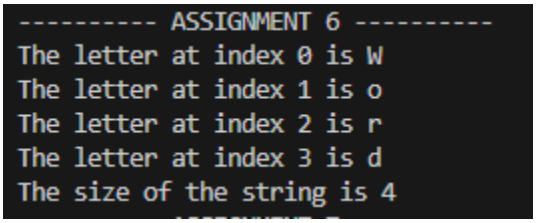
Assignment 5

In assignment 5, I used malloc to allocate memory on the heap. I first allocated space for a single integer and assigned it a value. Then, I allocated a block of memory large enough for five integer. I used a loop and pointer arithmetic to populate and print this array. Finally, I used the free command to release both blocks of memory back to the system to prevent memory leaks.

```
---------- ASSIGNMENT 5 ----------
The newly allocated memory holds the integer 5
Integer at index 0 is 1
Integer at index 1 is 2
Integer at index 2 is 3
Integer at index 3 is 4
Integer at index 4 is 5
```

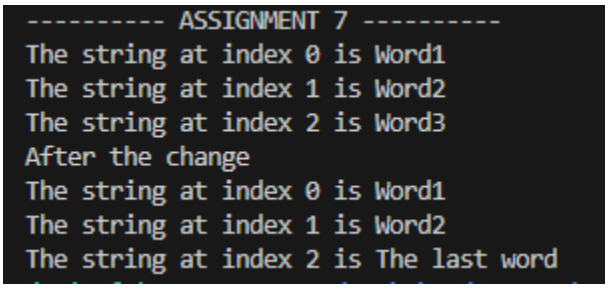
Assignment 6

In assignment 6, I used a character pointer (`char *ptr`) to point to a string literal. I iterated through the first four characters. Then I created a custom `str_length` function that takes a character pointer and uses a `while` loop to increment a counter until it reaches the null terminator (`\0`). The function then returns that value and I print it.



Assignment 7

In assignment 7, I created an array where each element is a pointer to a string (`char *ptr[]`). I used a loop with pointer notation (`*(ptr + i)`) to print each string in the list. I then modified the array by changing one of the pointers to look at a new string literal. Finally, I traversed the array again to confirm that the pointer now pointed to the updated text.



Link to github repository:

https://github.com/DMelkonyan111/Parallel-and-HPC-Homeworks