CS 2263

Lab3

Saheb Singh Arora: 3742233

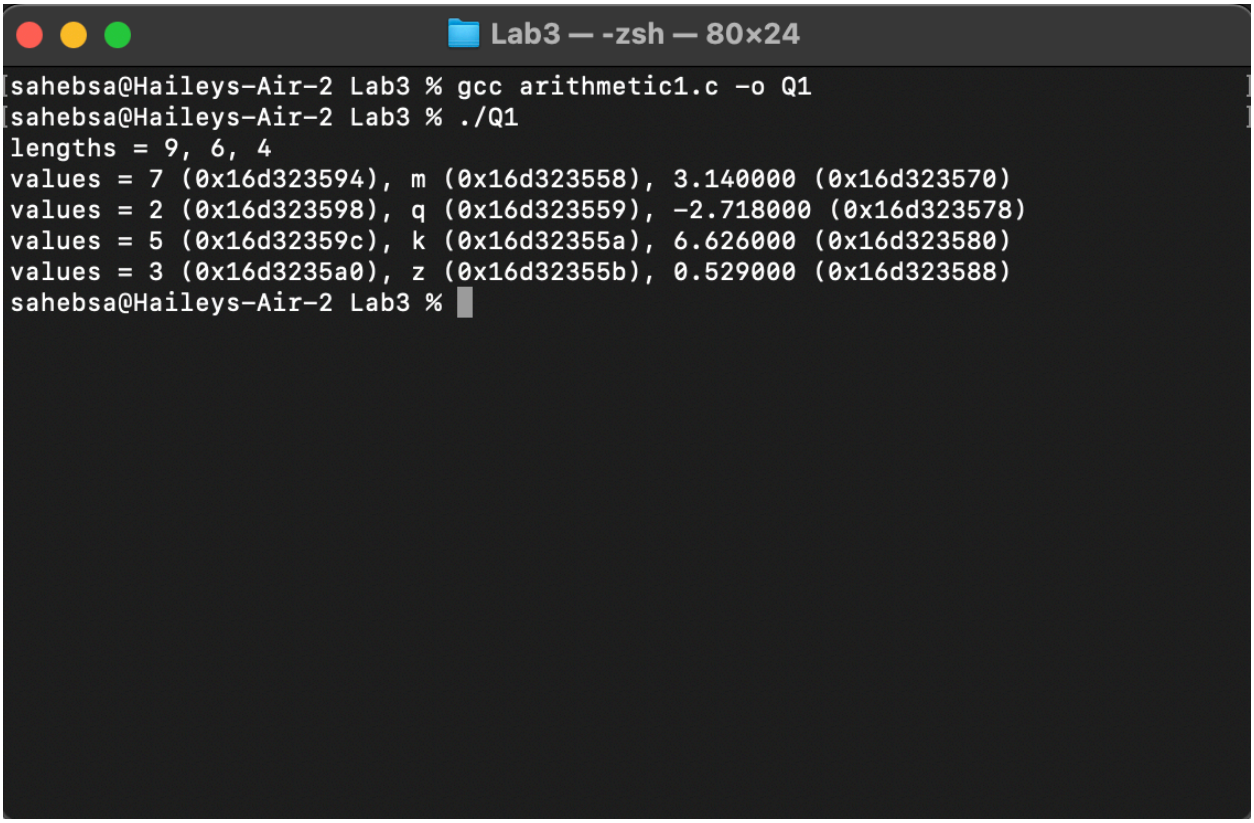
Exercise 1

#source code

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

int main(int argc, char *argv[]) {
    int arr1[] = {7, 2, 5, 3, 1, 6, -8, 16, 4};
    char arr2[] = {'m', 'q', 'k', 'z', '%', '>'};
    double arr3[] = {3.14, -2.718, 6.626, 0.529};
    int len1 = sizeof(arr1) / sizeof(int);
    int len2 = sizeof(arr2) / sizeof(char);
    int len3 = sizeof(arr3) / sizeof(double);
    printf("lengths = %d, %d, %d\n", len1, len2, len3);
    int *iptr = arr1;
    char *cptr = arr2;
    double *dptr = arr3;
    printf("values = %d (%p), %c (%p), %f (%p)\n", *iptr, (void
*)iptr, *cptr, (void *)cptr, *dptr, (void *)dptr);
    iptr++;
    cptr++;
    dptr++;
    printf("values = %d (%p), %c (%p), %f (%p)\n", *iptr, (void
*)iptr, *cptr, (void *)cptr, *dptr, (void *)dptr);
    iptr++;
    cptr++;
    dptr++;
    printf("values = %d (%p), %c (%p), %f (%p)\n", *iptr, (void
*)iptr, *cptr, (void *)cptr, *dptr, (void *)dptr);
    iptr++;
    cptr++;
    dptr++;
    printf("values = %d (%p), %c (%p), %f (%p)\n", *iptr, (void
*)iptr, *cptr, (void *)cptr, *dptr, (void *)dptr);
    return EXIT_SUCCESS;
}
```

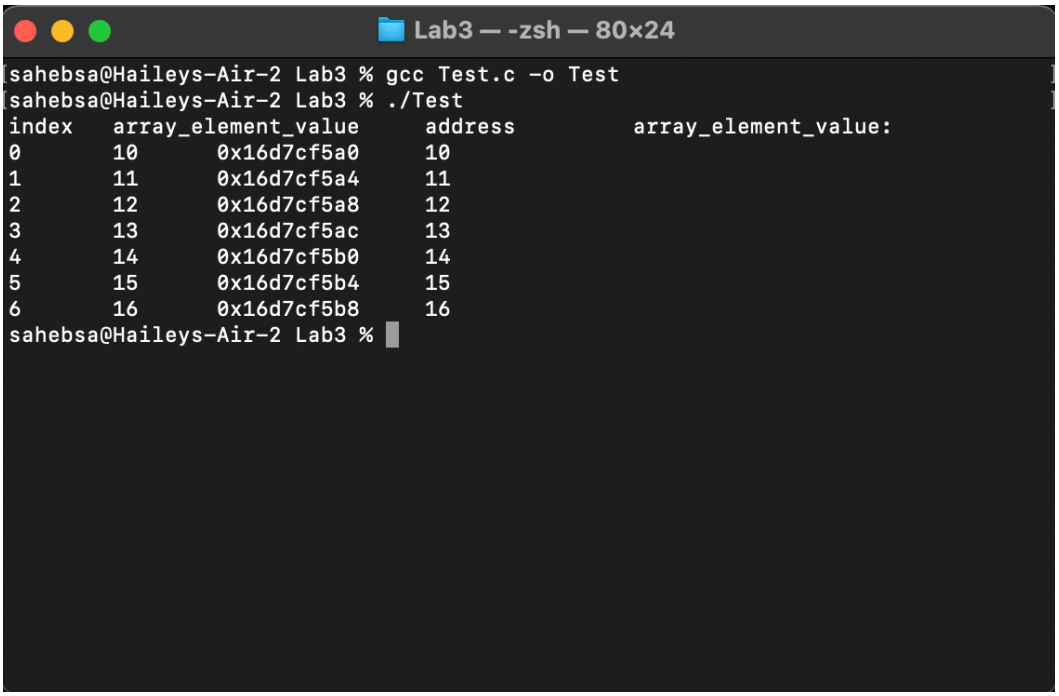
yes, in between consecutive print operations, the pointer variables iptr, cptr, and dptr are increased by 1.

No, various pointers have different increments. The size of the type to which the pointer is pointing determines the increment. The increments will change since iptr points to an int, cptr points to a char, and dptr points to a double. Pointer arithmetic in C is performed in terms of the pointed-to type's size. For instance, when a pointer is incremented to an int, the size of the int is added to the pointer's value, but when a pointer is incremented to a char, the size of the char is added.



Exercise 2

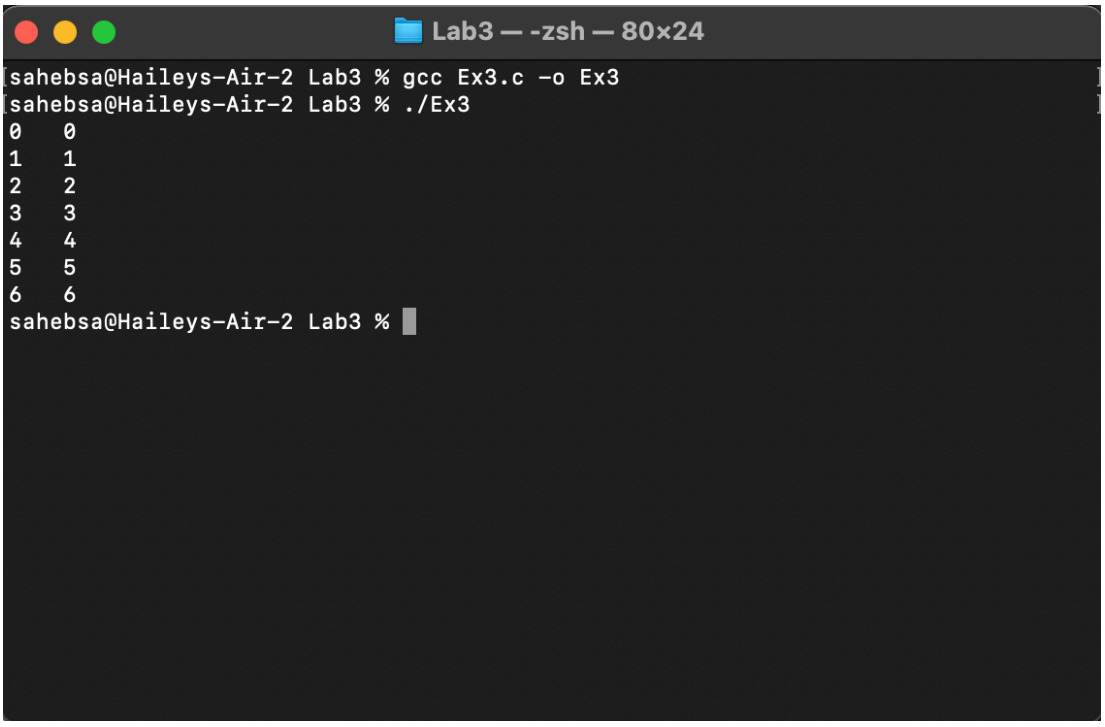
```
#include <stdio.h>

void printArrayTwice(int arr[], int size) {
    int *aptr = arr;


printf("index\tarray_element_value\taddress\t\tarray_element_val
ue:\n");
    for (int i = 0; i < size; i++) {
        printf("%d\t%d\t%p\t%d\n", i, arr[i], &arr[i], *aptr);
        aptr++;
    }
}

int main() {
    int arr[] = {10, 11, 12, 13, 14, 15, 16};
    int size = sizeof(arr) / sizeof(arr[0]);

    printArrayTwice(arr, size);

    return 0;
}
```



Exercise

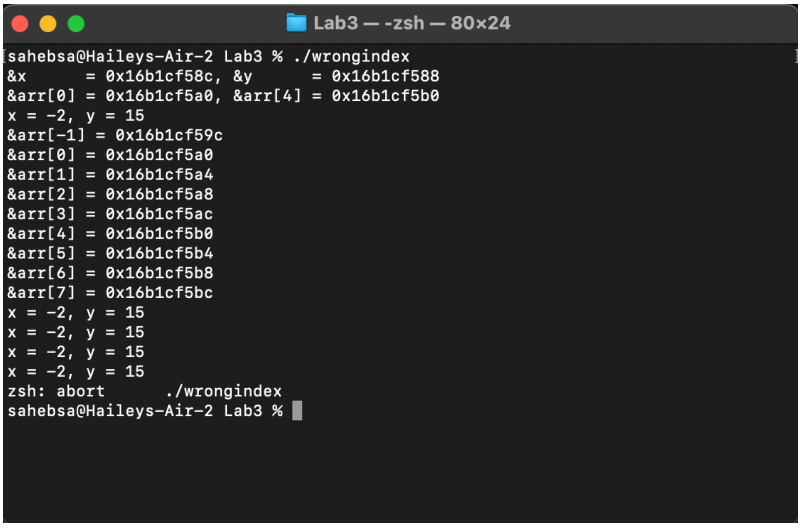
3

```c
#include <stdio.h>

int arrindex(int a[], int *p) {
    return p - a;
}

int main() {
    int arr[] = {10, 11, 12, 13, 14, 15, 16};
    int size = sizeof(arr) / sizeof(arr[0]);

    for (int i = 0; i < size; i++) {
        printf("%d   %d\n", i, arrindex(arr, &arr[i]));
    }

    return 0;
}
```



Exercise 4

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

int main(int argc, char *argv[]) {
    int x = -2;
    int arr[] = {0, 1, 2, 3, 4};
    int y = 15;

    printf("&x      = %p, &y      = %p\n", (void *)&x, (void
*)&y);
    printf("&arr[0] = %p, &arr[4] = %p\n", (void *)&arr[0],
(void *)&arr[4]);
    printf("x = %d, y = %d\n", x, y);

    for (int i = -1; i <= 7; i++) {
        printf("&arr[%d] = %p\n", i, (void *)&arr[i]);
    }

    arr[-1] = 7;
    printf("x = %d, y = %d\n", x, y);
    arr[5] = -23;
    printf("x = %d, y = %d\n", x, y);
    arr[6] = 108;
    printf("x = %d, y = %d\n", x, y);
    arr[7] = -353;
    printf("x = %d, y = %d\n", x, y);

    return EXIT_SUCCESS;
}
```



Stack Memory diagram

	A	B
1	Variable	Memory address
2		
3	x	0x16b1cf58c
4	y	0x16b1cf588
5	arr[-1]	0x16b1cf59c
6	arr[0]	0x16b1cf5a0
7	arr[1]	0x16b1cf5a4
8	arr[2]	0x16b1cf5a8
9	arr[3]	0x16b1cf5ac
10	arr[4]	0x16b1cf5b0
11	arr[5]	0x16b1cf5b4
12	arr[6]	0x16b1cf5b8
13	arr[7]	0x16b1cf5bc

Yes, there are a number of reasons why the numerical values output by the program might not match the numbers in the textbook. These include variances in the compiler and platform, memory layout discrepancies, and undefined behaviour brought on by accessing items with erroneous indices.