Pointers Labsheet

Ashwin R
AM.EN.U4CSE19343

Q 12. Does the following code run successfully to return 0 or does it generate a segmentation fault? If it runs fine, then what is the output? Otherwise explain why it segfaults. (A segmentation fault occurs when a program attempts to access a memory location that it is not allowed to access.)

```c
#include <stdio.h>
#include <stdlib.h>
void populate(int *a) {
    int *parray = malloc(2 * sizeof(int));
    parray[0] = 37;
    parray[1] = 73;
    a = parray;
}
int main() {
    int *a = NULL;
    populate(a);
    printf("a[0] = %d and a[1] = %d\n", a[0], a[1]);
    return 0;
}
```

Ans:

A pointer can not be compared to another pointer, i.e the address stored by one pointer can't be accessed by another different pointer (unless it's a pointer to a pointer). So in the populate function we can't give the pointer 'a' access to the address in pointer 'parray'. So in the main function the a[0] and a[1] is not declared or has no memory address so the program will give a segmentation fault. If we remove parray and dynamically allocate a then we will get the required output.

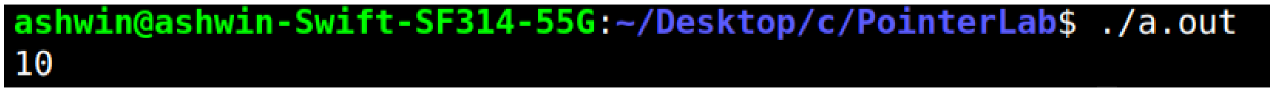
Modified code:

```
#include <stdio.h>
#include <stdlib.h>
void populate(int *a) {
    a[0] = 37;
    a[1] = 73;
}
int main() {
    int *a = NULL;
    a=malloc(2*sizeof(int));
    populate(a);
    printf("a[0] = %d and a[1] = %d\n", a[0], a[1]);
    return 0;
}
```

Q 13. Write a basic program with pointers as directed below.

a. Declare a pointer to an integer variable ptr.
   `int * ptr;`
b. Use malloc to dynamically allocate memory for ptr
   `ptr = (int *) malloc( sizeof(int) );`
c. Assign an integer value to the memory pointed to by ptr.
   `*ptr = 10;`
d. Print the value pointed to by ptr to the terminal
   `printf(“%d\n”, *ptr);`
e. Free the ptr

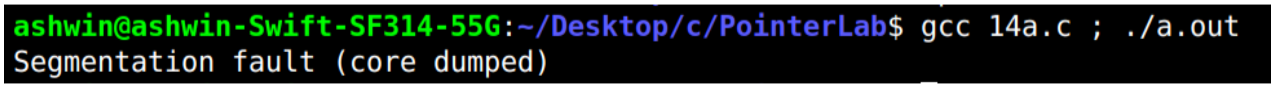
OUTPUT: **10**



```
ashwin@ashwin-Swift-SF314-55G:~/Desktop/c/PointerLab$ ./a.out
10
```

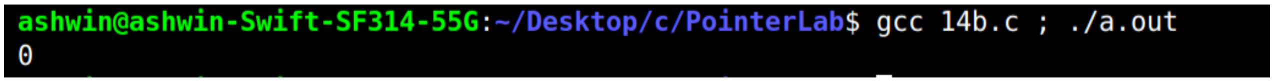
Q 14. Perils of pointers: Variations of Q13 to simulate the problems due to mishandling of pointers. (All programs attached)

a. **A case of null pointer**



```
ashwin@ashwin-Swift-SF314-55G:~/Desktop/c/PointerLab$ gcc 14a.c ; ./a.out
Segmentation fault (core dumped)
```

b. **Another case of null ptr**



```
ashwin@ashwin-Swift-SF314-55G:~/Desktop/c/PointerLab$ gcc 14b.c ; ./a.out
0
```

c. **A case of memory leak**
   The system will crash since the heap memory will get filled because of not freeing the dynamically allocated memory

d. **A case of not allowing memory leak**
   The program will be in an infinite loop, but won't crash the system as the memory is freed in every iteration

e. **A case of lost pointer and memory leak**
   i. The access to location_1 is lost. It is impossible to retrieve the value 5.
   ii. It can't be freed either since the pointer is lost. This leads to memory leak.

f. **Another case of lost pointer:**
   i. The access to location_1 is lost. It is impossible to retrieve the value 5.
   ii. It can't be freed either since the pointer is lost. This leads to memory leak.