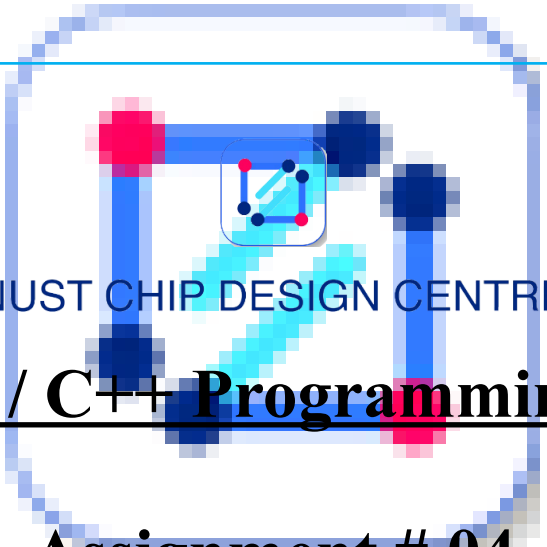


C / C++ Programming

Assignment # 04
Pointers

Name	*Muddassir Ali Siddiqui*
Instructor	*Miss Hira Sohail*
Date	*15th July 2025*

1. In-Lab Tasks: (*Write your lab task & screenshots here*)

i. Task 1:

Question: How would we initialize p so that it points to x?
Answer:

```
int x;
int *p = &x;
```

Question: What would happen if we passed p instead of *p to printf?
Answer: It prints the address of x.
Question: What would happen if we dereferenced a pointer that had the value of NULL?
Answer: It crashes the program due to segmentation error.

ii. Task 2:

DO THIS: Copy the previous code into your file and test it. Then, add more lines of code to do the following:

1. Update t to point to c. Use a pointer dereference to change the value of c to 555. Verify that it worked by adding a printout. Does this change any of the other values?

lectures > c > Assignments > week 1 > assignment4 > muddassir_ali_siddiqui_Assignmen2 > codes > C ass4task.c > ...

```c
#include <stdio.h>

int main (){
    int a = 42;
    int b = 7;
    int c = 999;

    int *t = &c;    // Update t to point c
    int *u = NULL;

    printf("%d %d\n", a, *t);

    c = b;
    u = t;
    printf("%d %d\n", c, *u);

    a = 8;
    b = 8;
    printf("%d %d %d %d\n", b, c, *t, *u);

    *t = 555;   //Use a pointer dereference to change the value of c to 555
    printf("%d %d %d %d %d\n", a, b, c, *t, *u);

    return 0;
}


// int *p = NULL;
    // int x = 5;
    // p = &x;
    // printf("Pointer value of p is %p\n", p);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS TEROSHDL: LOG REPORT TEROSHDL: TIMING

```
[cc@ncdc-0053 codes]$ ./ass4task
Pointer value of p is 0x7ffcf59afdf4
Dereferenced value of p is 5, and x = 5
[cc@ncdc-0053 codes]$ gcc ass4task.c -o ass4task
[cc@ncdc-0053 codes]$ ./ass4task
42 42
7 42
8 7 8 8
123 8 7 123 123
[cc@ncdc-0053 codes]$ gcc ass4task.c -o ass4task
[cc@ncdc-0053 codes]$ ./ass4task
42 999
7 7
8 7 7 7
8 8 555 555 555
[cc@ncdc-0053 codes]$
```

2. Change the value of c again using a direct assignment. Verify that the pointer t still points to the value by printing the result of dereferencing it.

lab4_task2.c | lab4_task3.c | lab4_task4.c | lab4_task6.c | ass4task.c

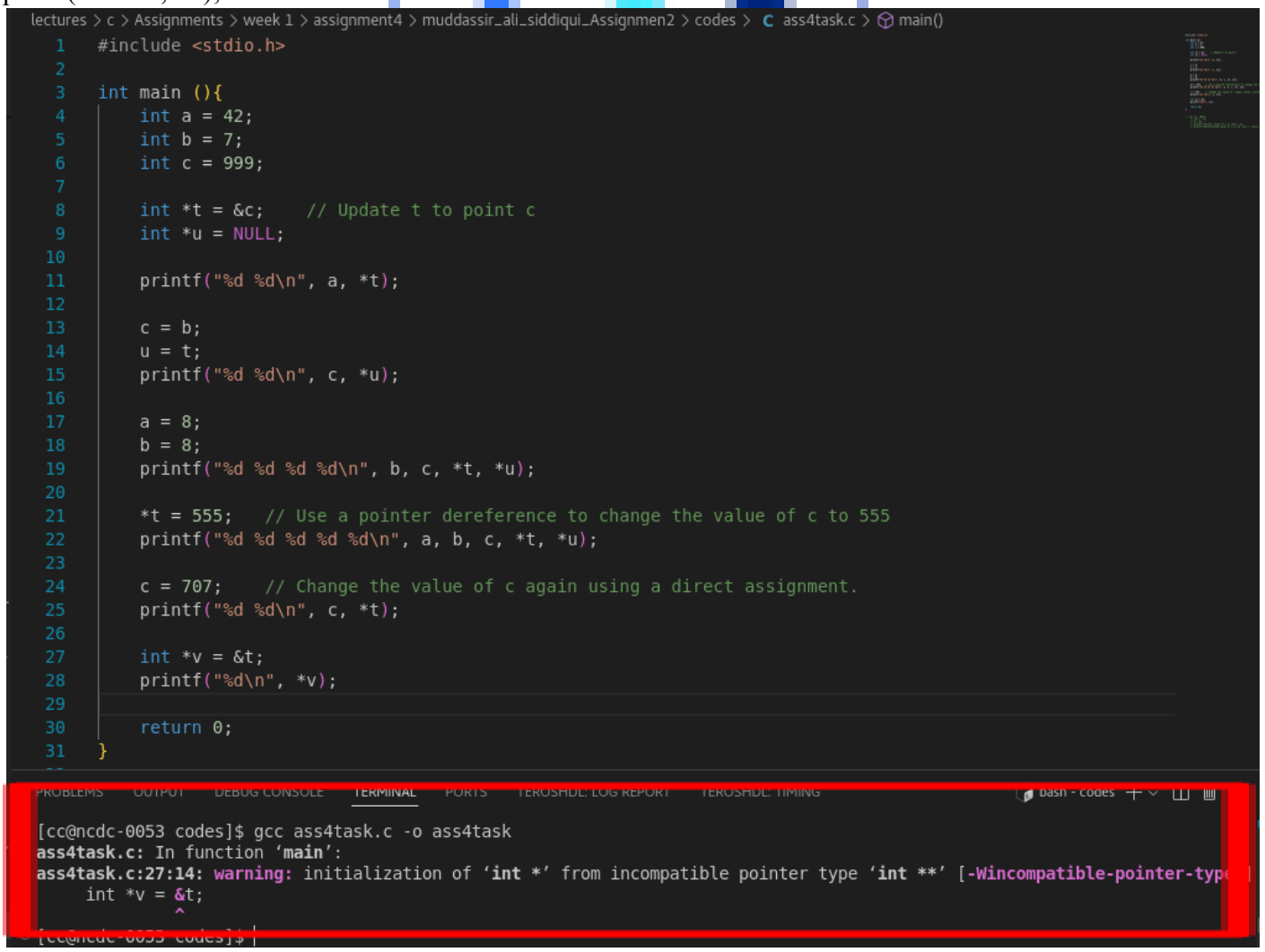
lectures > c > Assignments > week 1 > assignment4 > muddassir_ali_siddiqui_Assignmen2 > codes > ass4task.c > main

```c
#include <stdio.h>

int main (){
    int a = 42;
    int b = 7;
    int c = 999;

    int *t = &c;    // Update t to point c
    int *u = NULL;

    printf("%d %d\n", a, *t);

    c = b;
    u = t;
    printf("%d %d\n", c, *u);

    a = 8;
    b = 8;
    printf("%d %d %d %d\n", b, c, *t, *u);

    *t = 555;   // Use a pointer dereference to change the value of c to 555
    printf("%d %d %d %d %d\n", a, b, c, *t, *u);

    c = 707;    // Change the value of c again using a direct assignment.
    printf("%d %d\n", c, *t);

    return 0;
}
```

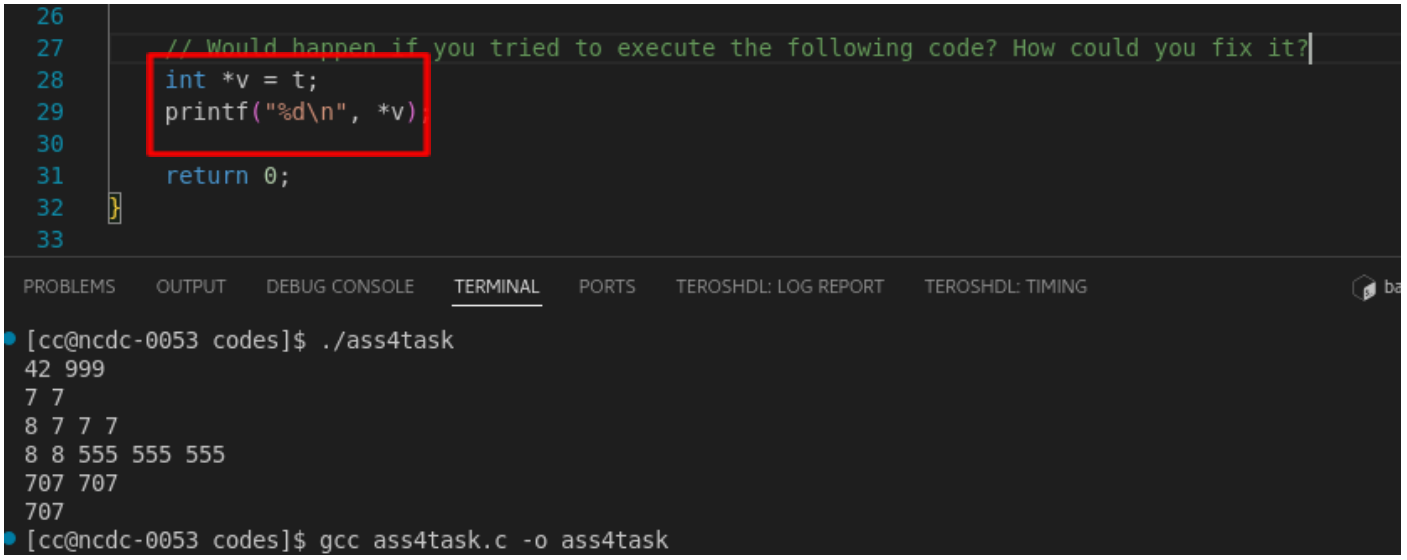
PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | TEROSHDL: LOG REPORT | TEROSHDL: TIMING

```
[cc@ncdc-0053 codes]$ ./ass4task
42 42
7 42
8 7 8 8
123 8 7 123 123
[cc@ncdc-0053 codes]$ gcc ass4task.c -o ass4task
[cc@ncdc-0053 codes]$ ./ass4task
42 999
7 7
8 7 7 7
8 8 555 555 555
[cc@ncdc-0053 codes]$ gcc ass4task.c -o ass4task
[cc@ncdc-0053 codes]$ ./ass4task
42 999
7 7
8 7 7 7
8 8 555 555 555
707 707
[cc@ncdc-0053 codes]$
```

Question: Would happen if you tried to execute the following code? How could you fix it?
int *v = &t;
printf("%d\n", *v);



Answer: In this code integer pointer points the pointer which is not not either it should be point the value of t like in this:



Or v should be pointer pointer to point the pointer t like:

```c
int main (){
    int a = 42;
    int b = 7;
    int c = 999;

    int *t = &c;    // Update t to point c
    int *u = NULL;

    printf("%d %d\n", a, *t);

    c = b;
    u = t;
    printf("%d %d\n", c, *u);

    a = 8;
    b = 8;
    printf("%d %d %d %d\n", b, c, *t, *u);

    *t = 555;   // Use a pointer dereference to change the value of c to 555
    printf("%d %d %d %d %d\n", a, b, c, *t, *u);

    c = 707;    // Change the value of c again using a direct assignment.
    printf("%d %d\n", c, *t);

    // Would happen if you tried to execute the following code? How could you fix it?
    int **v = &t;
    printf("%d\n", *v);

    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  TEROSHDL: LOG REPORT  TEROSHDL: TIMING

[cc@ncdc-0053 codes]$ ./ass4task
42 999
7 7
8 7 7 7
8 8 555 555 555
707 707
707
[cc@ncdc-0053 codes]$ gcc ass4task.c -o ass4task
[cc@ncdc-0053 codes]$ ./ass4task
42 999
7 7
8 7 7 7
8 8 555 555 555
707 707
1052345828
[cc@ncdc-0053 codes]$
```

The result remains the same.

This illustrates an important concept: pointers can point to almost anything, even other pointers!