**Operating Systems**

**LAB # 03**



**Spring 2022**

**CSE204L**

Submitted by: **Maaz Habib, Alishba Orakzai, Nida Asmat Burki**

Registration No: **20PWCSE1952, 20pwcse1953, 20pwcse1955**

Class Section: **C**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Mian Ibad Ali Shah**

tuesday, March 26, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Introduction to C Programming**

**Objective:**

 To gain experience with

1. Writing simple c programs with more than one function (Parameters passed by value)

2. Basic concepts of Pointers in C

3. Passing parameters to the function by pointers.

4. Using Arrays in C

5. Using Structures in C

6. Use of Linked List

**Purpose:**

To gain experience about C programming

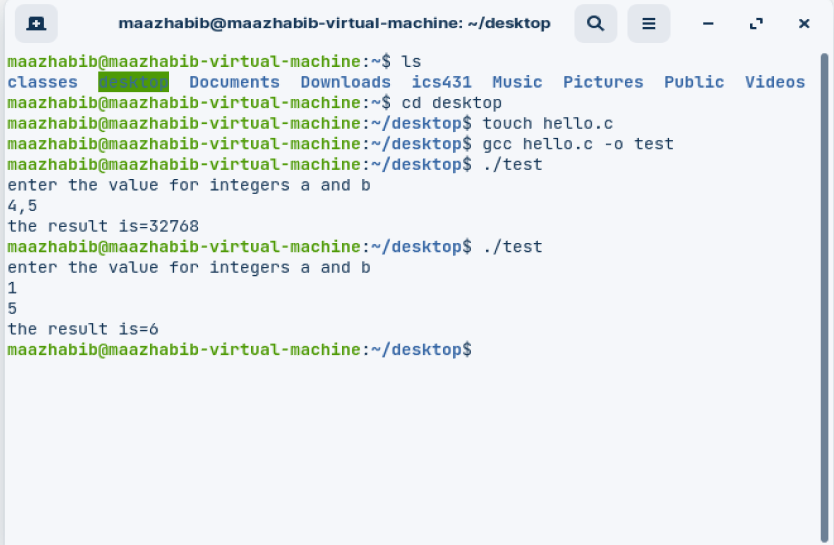
**1. A Simple C program with more than one function (Parameters passed by value)**

 The following program reads two numbers and finds the sum of those two numbers

**Program**



**Output**



**2. Basic concepts of Pointers in C**

Every variable in C has a name (variable name), a memory location (to store the data), and an address.

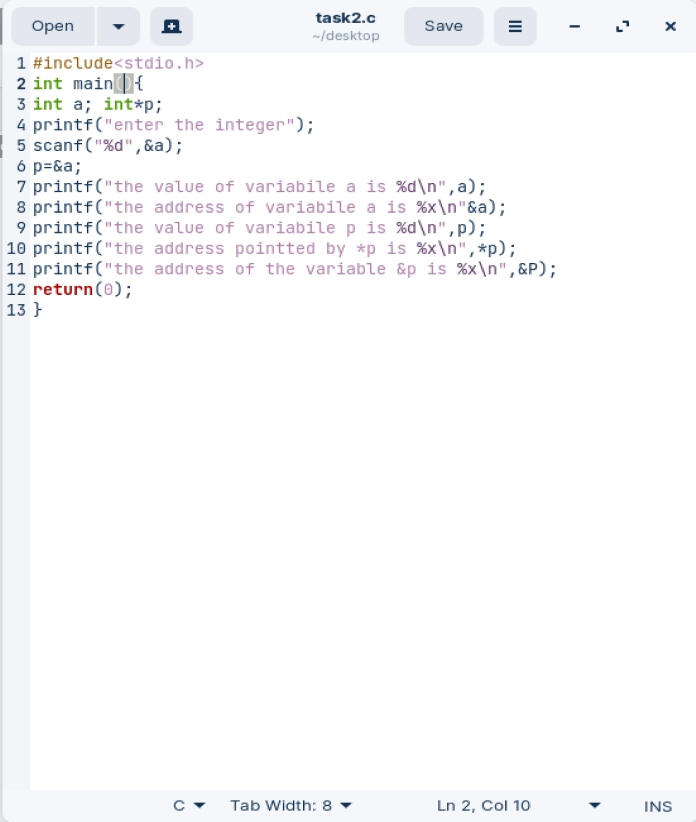
For the variable declaration:   int  num; num----------&gt;variable name

Memory

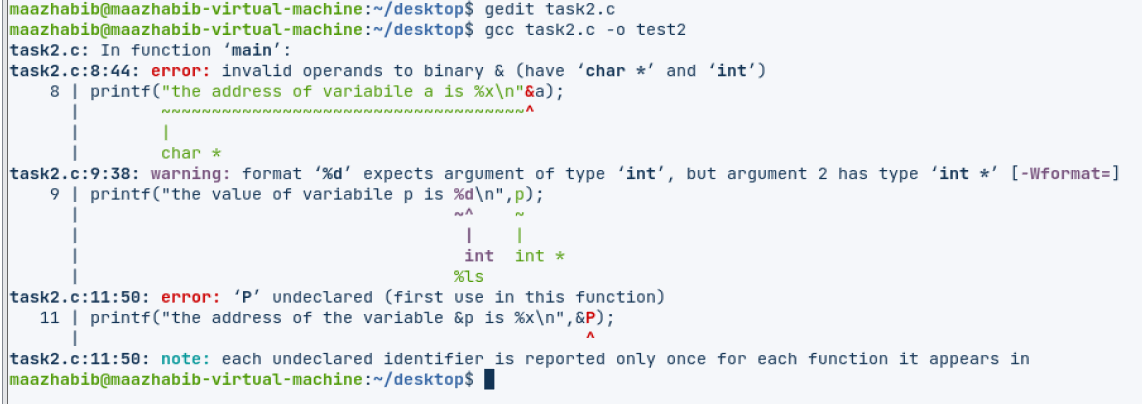
 8152       ------------------&gt;   address

A variable used to store the address value is called as the Pointer. It can be defined as int \*ptr; The following program demonstrates about the pointer variable, \*  and &amp; operators.

**Program:**



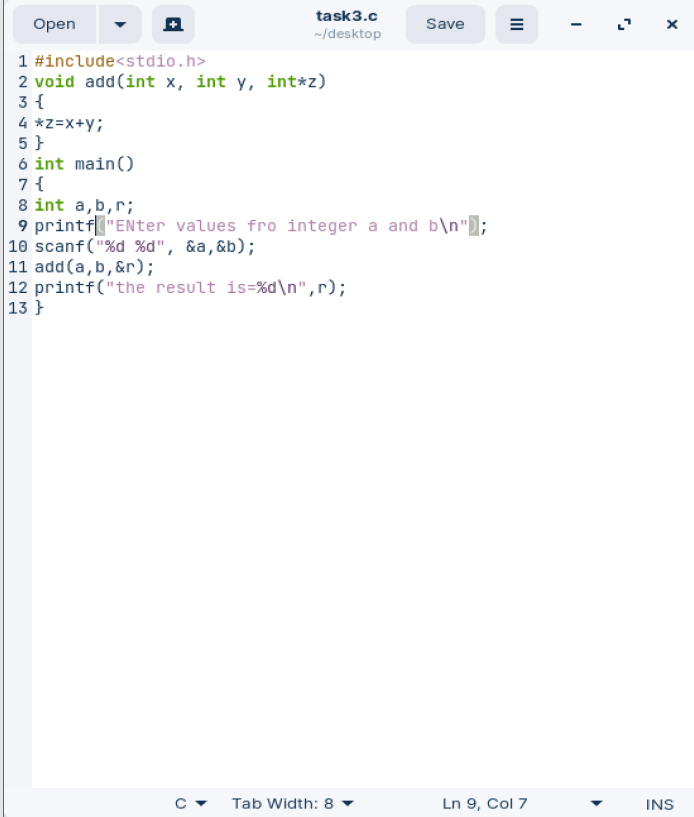
**Program:**



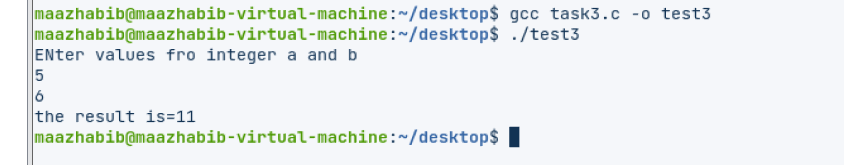
**3. Passing parameters to function by pointers**

         The following program reads two values and adds the two numbers, the result is passed by pointer.

**Program**



**Output**



**4. Using Arrays in C**

Example:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

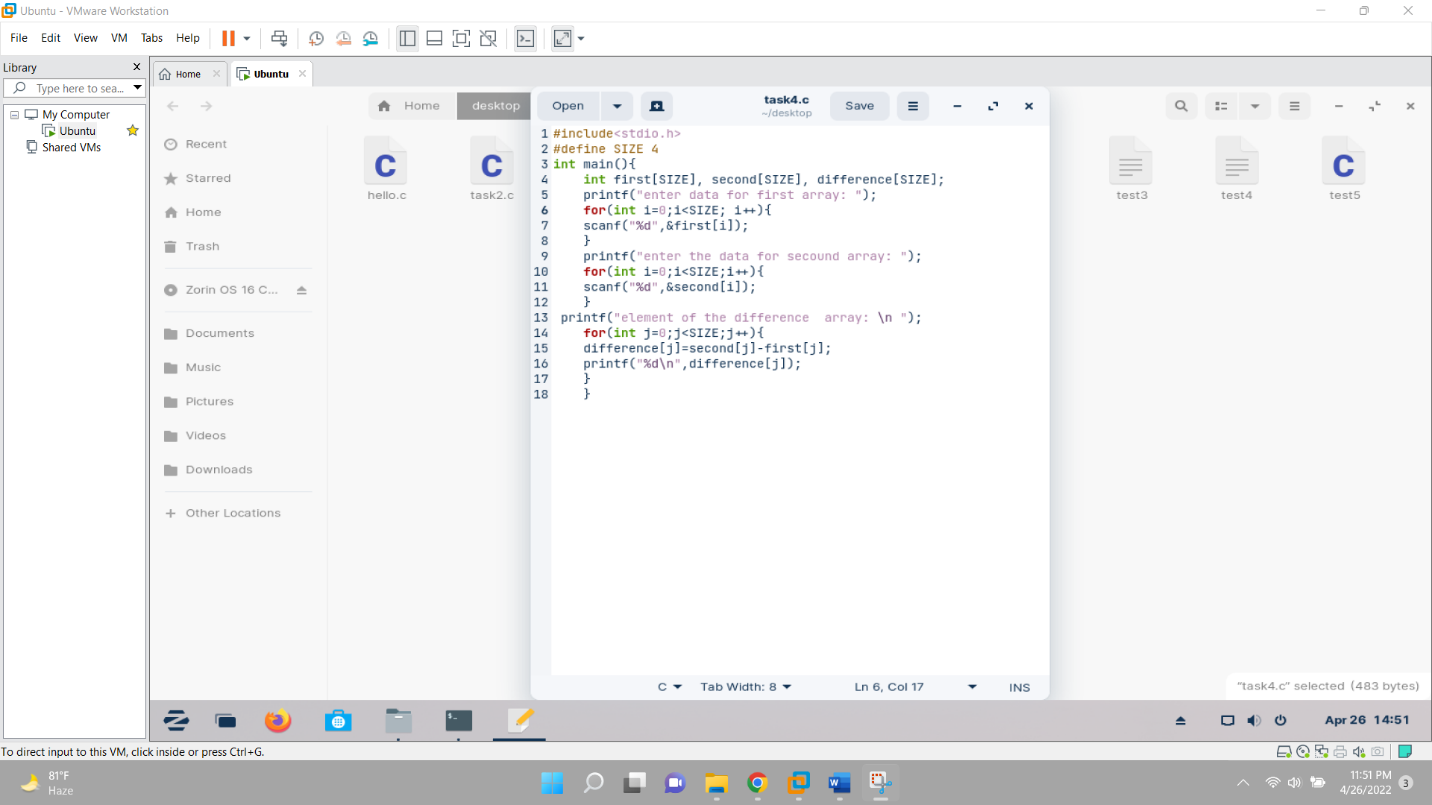
Create three arrays. Read data into the first two of them. Subtract each

element in the first array from the corresponding element in the second

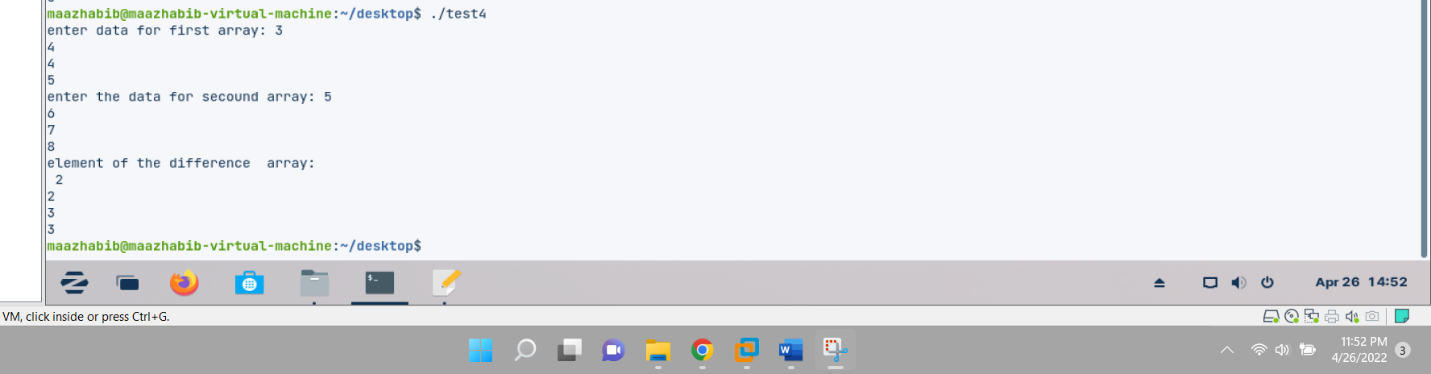
array. Store the differences in the third array. Print all the arrays.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Program:**

****

**Output**

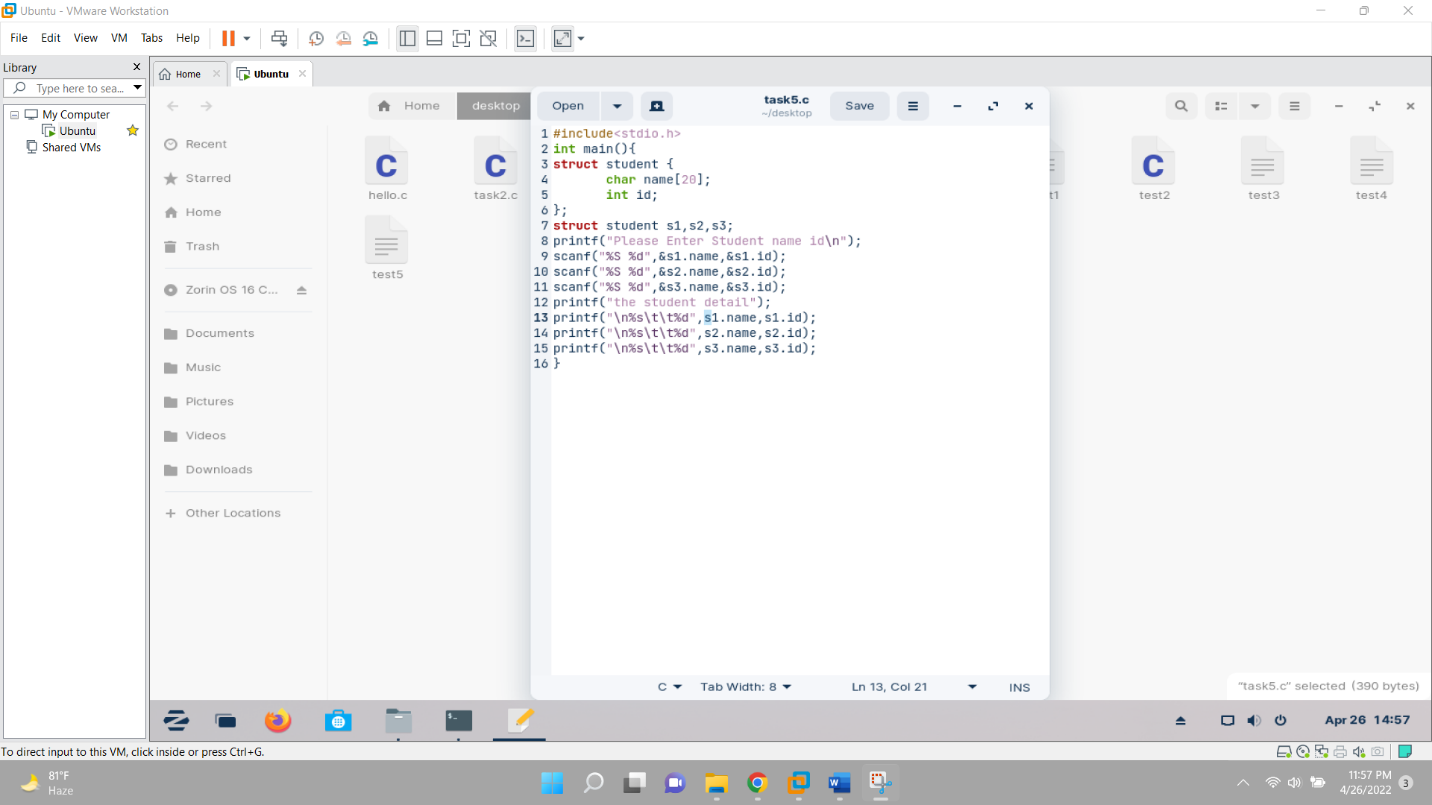
****

**5. Using Structures in C**

**Structures**

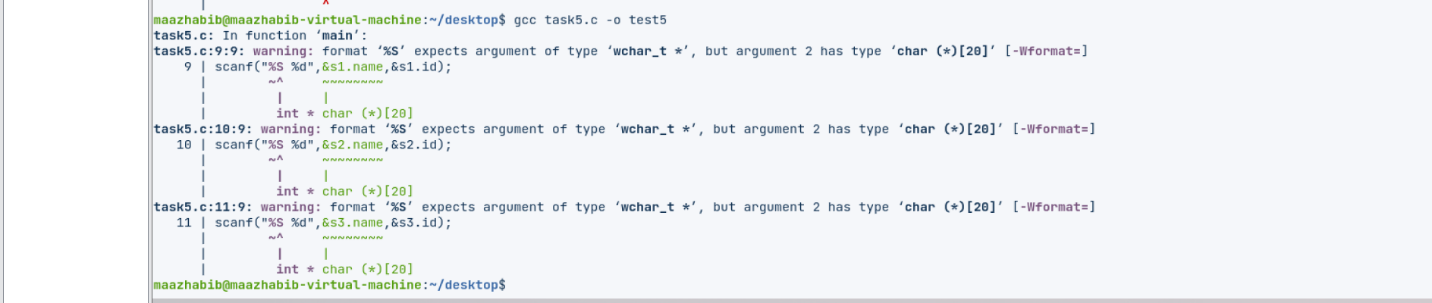
 Structure is a collection / group of different / same variables.

**Program**

****

**#icpcp**

**Output**

****

**6. Use of Linked List in C**

Using malloc to obtain memory at run-time.

* Memory can be allocated dynamically (at run-time) using the function malloc() – accessible through &lt;stdlib.h&gt;
* The allocation is made from a special memory area called the heap.
* The function, malloc() returns a pointer (address) to the allocated storage.
* However, malloc() does not associate any type to the pointer it returns – it is said to be void. For the pointer to be useful, it must be associated with a type using casting.

e.g.

int \*int\_ptr;

int\_ptr=(int \*) malloc(2);

\*int\_ptr =17;

The above statements reserve two bytes and return the address of first byte, cast it to int and assign it to integer pointer int\_ptr.

* Since the bytes allocated to int is system-dependent, it is safer to use the function sizeof () to get the actual number of bytes associated with the particular type being considered.
* sizeof() is system-independent and can be used even with user-defined types.

 Thus, the above statements are better represented as follows:

                        int \*int\_ptr;

int\_ptr=(int \*) malloc(sizeof(int));

\*int\_ptr =17;

* Note that there is no name associated with the memory obtained by malloc. It can

only be accessed as \*int\_ptr. It is sometimes called anonymous variable.

* Thus, should int\_ptr be given another address, the location (returned by malloc) will be lost . It can neither be accessed by the program nor by the system. It is said to be a lost object.

When we no longer need a dynamic variable, we can return the storage it occupies using the free() function.

e.g. free(int\_ptr);

**7. Exercises**

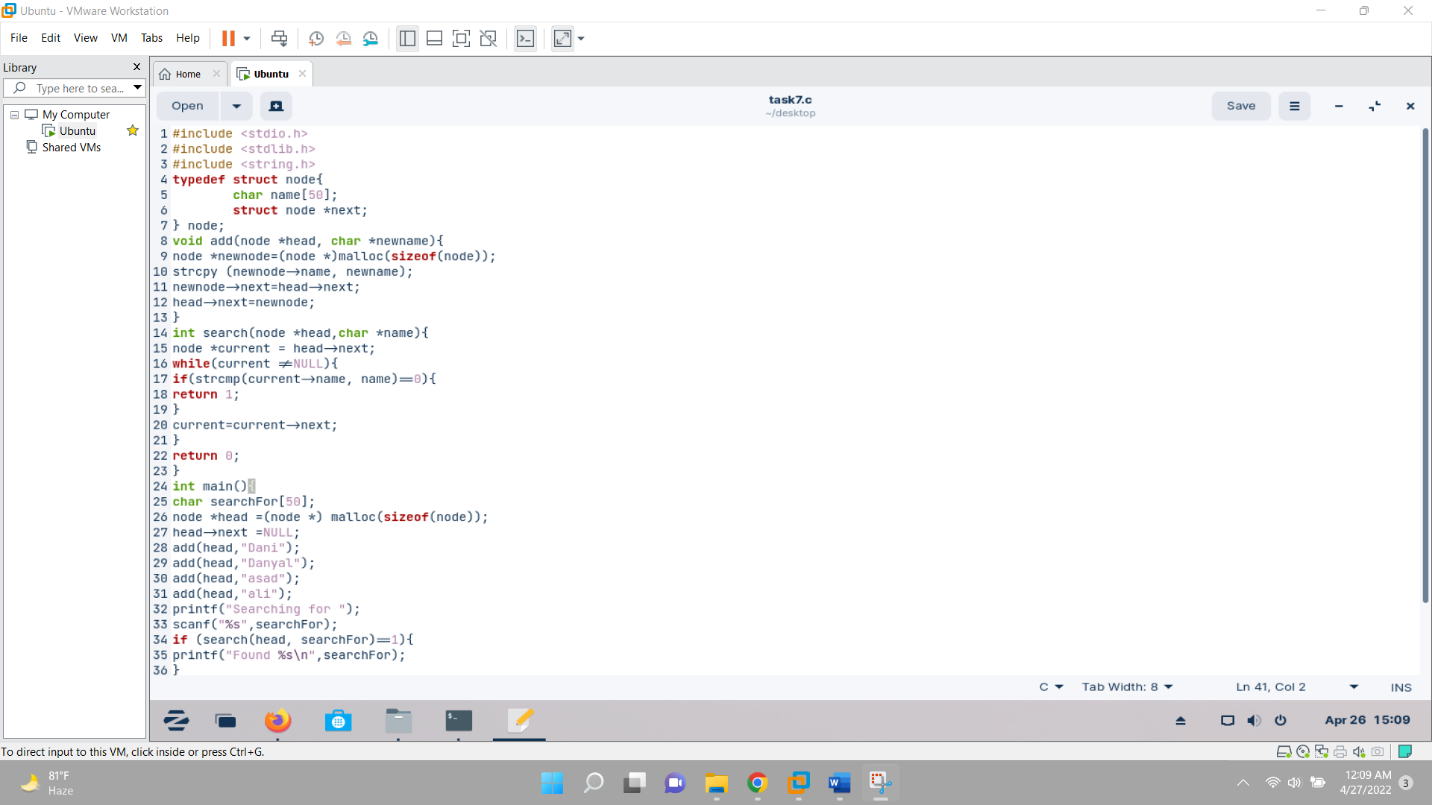
1. Run all the above example problems and try to understand the concepts.

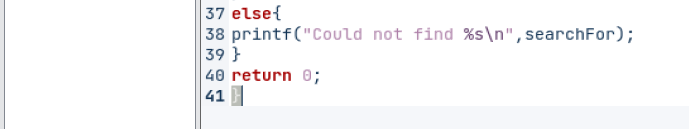
2. Write a complete menu driven program to do the following:

* Build a linked list to save a list of names. Name will not exceed 50 characters.
* Write a function add to append a new name to the list. The function prototype is given as void add (list \*head, char \*newname);
* Write a function search to look for a given name in the list. If that name is found in list then the function should return true, otherwise, return false.
* Write a main method to test your two functions.

            In C language, the boolean type and the boolean literals (true, false) are not defined. We can define these in our program as follow: typedef enum {false = 0, true} bool The skeleton of your program should look like the following:

**Program**

****

****

**Output**

