**COMMAND LINE ARGUMENT, VARIABLE ARGUMENT HANDLING**

Write a program to

a. read a name(of max length 40 characters), ip address (as char \* string in dotted notation) and port number (unsigned short) of the cloud server as command line arguments.

**A computer screen shot of a program

Description automatically generated**

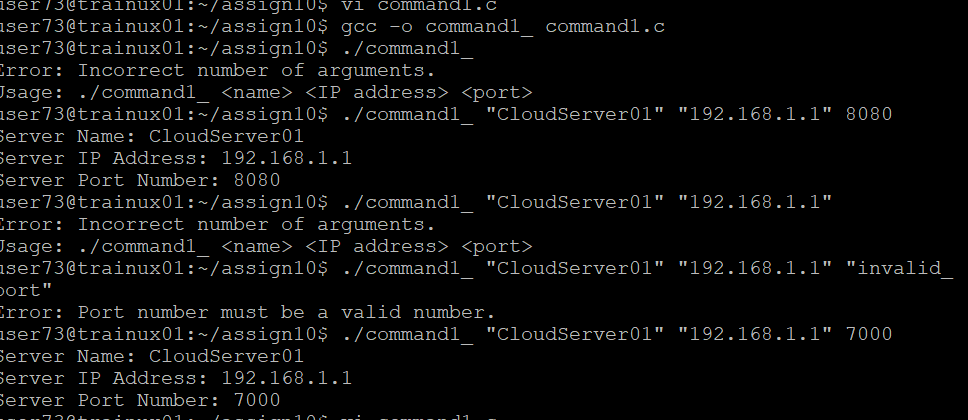
**A screen shot of a computer code

Description automatically generated**

**b. Validate if the required number of arguments have been received before proceeding. Else report error and return.**

* Validation of Argument Count
* Port Number Validation





**c. Validate every argument received for valid range of values. [Refer ip address range, port range to do validations]**

**IP Address Validation (isValidIP):**

* The function isValidIP checks that the IP address has exactly four segments, separated by periods (.).
* Each segment is checked to ensure it is a valid number between 0 and 255.
* The function returns 1 if the IP is valid, otherwise 0.

**Port Validation (isValidPort):**

* The function isValidPort checks whether the port number is within the valid range (1 to 65535).
* It returns 1 if valid, 0 if not.

**d. Store the values in a data structure and display using a function passing data structure**

**void display(struct server \*servercfg);**

**A computer screen shot of a program code

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**e. Implement a function update() to prompt user, to modify all the server attributes and to display the updated configuration.**

**// to read, update configuration and return status as SUCCESS/FAILURE**

**Int update(struct server \*servercfg);**

**A screenshot of a computer

Description automatically generated**

**f. Specify atleast 6 test cases (positive and negative ) to test command line inputs and update operations**

**Test Case 1:** Valid Command Line Input (Positive)

**Test Case 2**: Invalid IP Address Format (Negative)

**Test Case 3:** Port Number Out of Range (Negative)

**Test Case 4:** Server Name Exceeds Maximum Length (Negative)

**Test Case 5:** Successful Update with Valid Inputs (Positive)

**Test Case 6:** User Cancels Update (Negative)

**g. Check for memory leaks and fix them.**

**Valgrind:** To check for memory leaks in the program

free(servercfg.name); // Free allocated memory

free(servercfg.ipAddress);

// Free dynamically allocated memory before exiting

free(servercfg.name);

free(servercfg.ipAddress);

**2. Implement a log() with signature as below to display all the input arguments as per their type. [Hint: In log() , use vfprintf() to display the received inputs]**

**void log(const char \*format, …);**

**For e.g.**

**int main()**

**{**

**int count = 10;**

**char prefix = ‘h’;**

**char label[] = “India”;**

**…**

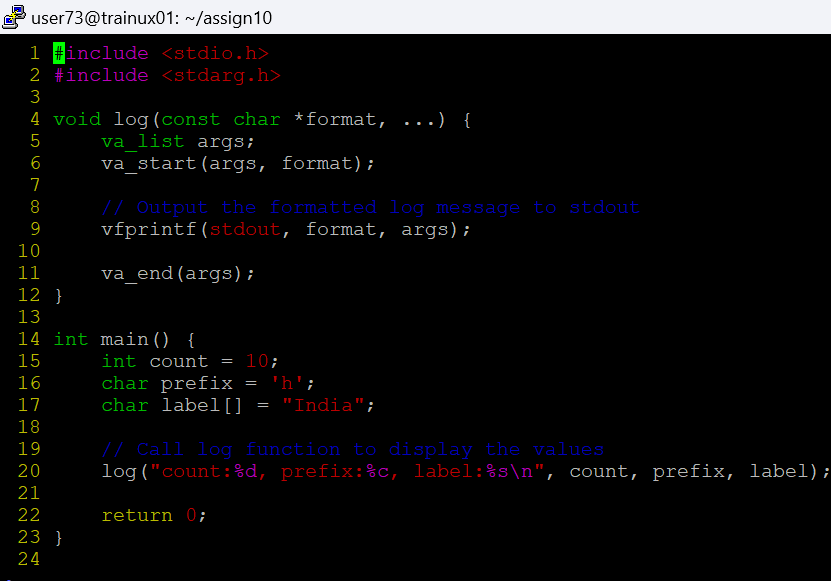
**log(“count:%d, prefix:%c, label:%s”, count, prefix, label);**

**…**

**}**

**Expected Output:**

**count:10,prefix:h,label:India**

****

**A black background with white text

Description automatically generated**

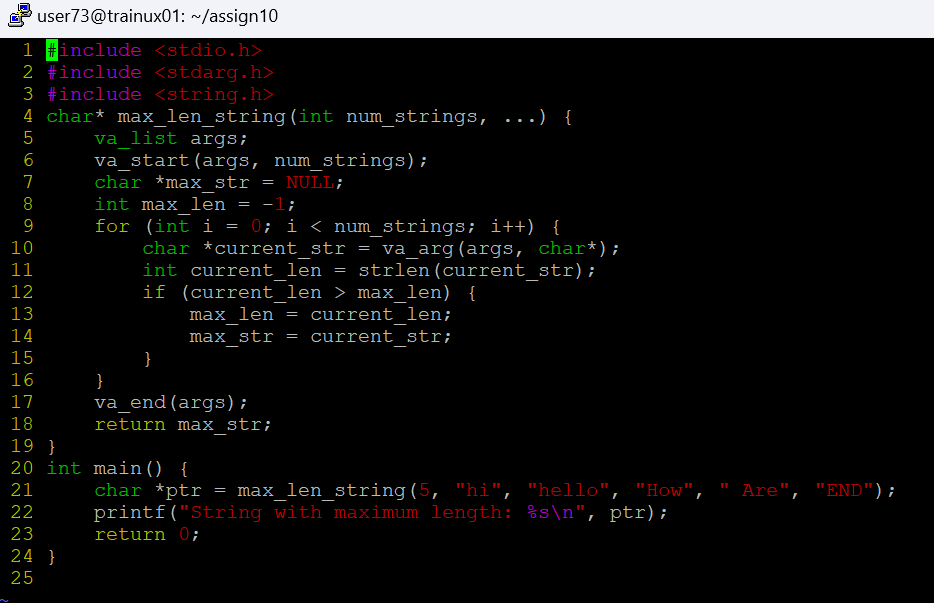
**3. Refer the code “find\_max.c”. Add a function below to accept variable number of strings and to return the string with maximum length to the caller. In case of strings with same length, return the first string in the input sequence**

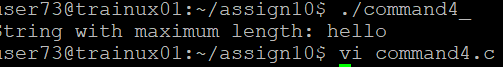
**max\_len\_string(<variable number of arguments>)**

**Eg. Code below shoud output “hello”**

**char \*ptr = max\_len\_string(“hi”, “hello”, “How”, “ Are”, “END”);**

**printf(“%s”, ptr);**

****

****