# Command Line Argument, Variable Argument Handling Assignment

1. Write a program to
   1. 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.
   2. Validate if the required number of arguments have been received before proceeding. Else report error and return.
   3. Validate every argument received for valid range of values.

[Refer [ip address range](https://en.wikipedia.org/wiki/IP_address#:~:text=IPv4%20addresses%20are%20usually%20represented,%2C%20e.g.%2C%20172.16.254.1.), [port range](https://www.sciencedirect.com/topics/computer-science/registered-port#:~:text=Port%20numbers%20can%20run%20from,application%20processes%20on%20other%20hosts.) to do validations]

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

void display(struct server \*servercfg);

* 1. 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);

* 1. Specify atleast 6 test cases (positive and negative ) to test command line inputs and update operations
  2. Check for memory leaks and fix them.

**Ans:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_NAME\_LENGTH 40

#define MAX\_IP\_LENGTH 16

struct server {

char name[MAX\_NAME\_LENGTH]; // Server name

char ip[MAX\_IP\_LENGTH]; // IP address in dotted notation

unsigned short port; // Port number

};

void display(struct server \*servercfg);

int update(struct server \*servercfg);

int is\_valid\_ip(const char \*ip);

int is\_valid\_port(unsigned short port);

int main(int argc, char \*argv[]) {

if (argc != 4) {

printf("Error: Invalid number of arguments.\n");

printf("Usage: %s <name> <ip\_address> <port>\n", argv[0]);

return 1;

}

struct server servercfg;

// Read and validate name

if (strlen(argv[1]) >= MAX\_NAME\_LENGTH) {

printf("Error: Name length exceeds the maximum allowed (40 characters).\n");

return 1;

}

strncpy(servercfg.name, argv[1], MAX\_NAME\_LENGTH - 1);

servercfg.name[MAX\_NAME\_LENGTH - 1] = '\0';

// Read and validate IP address

if (!is\_valid\_ip(argv[2])) {

printf("Error: Invalid IP address.\n");

return 1;

}

strncpy(servercfg.ip, argv[2], MAX\_IP\_LENGTH - 1);

servercfg.ip[MAX\_IP\_LENGTH - 1] = '\0';

// Read and validate port number

unsigned short port = (unsigned short)atoi(argv[3]);

if (!is\_valid\_port(port)) {

printf("Error: Invalid port number. It should be between 1 and 65535.\n");

return 1;

}

servercfg.port = port;

// Display the initial configuration

display(&servercfg);

// Prompt user to update the server configuration

if (update(&servercfg) == 1) {

display(&servercfg);

} else {

printf("Update failed.\n");

}

return 0;

}

// Function to display server configuration

void display(struct server \*servercfg) {

printf("\nServer Configuration:\n");

printf("Name: %s\n", servercfg->name);

printf("IP Address: %s\n", servercfg->ip);

printf("Port: %u\n", servercfg->port);

}

// Function to update server configuration

int update(struct server \*servercfg) {

char input[MAX\_NAME\_LENGTH];

char ip[MAX\_IP\_LENGTH];

unsigned short port;

// Prompt for server name

printf("\nEnter new server name (max 40 chars): ");

fgets(input, MAX\_NAME\_LENGTH, stdin);

input[strcspn(input, "\n")] = '\0'; // Remove newline character

if (strlen(input) > 0) {

strncpy(servercfg->name, input, MAX\_NAME\_LENGTH - 1);

servercfg->name[MAX\_NAME\_LENGTH - 1] = '\0';

}

// Prompt for IP address

printf("Enter new IP address: ");

fgets(ip, MAX\_IP\_LENGTH, stdin);

ip[strcspn(ip, "\n")] = '\0'; // Remove newline character

if (is\_valid\_ip(ip)) {

strncpy(servercfg->ip, ip, MAX\_IP\_LENGTH - 1);

servercfg->ip[MAX\_IP\_LENGTH - 1] = '\0';

} else {

printf("Invalid IP address format.\n");

return 0;

}

// Prompt for port number

printf("Enter new port number (1-65535): ");

if (scanf("%hu", &port) != 1 || !is\_valid\_port(port)) {

printf("Invalid port number.\n");

return 0;

}

servercfg->port = port;

return 1; // Success

}

// Function to validate IP address format (in dotted notation)

int is\_valid\_ip(const char \*ip) {

int dots = 0;

int num;

int i = 0;

while (\*ip) {

if (isdigit(\*ip)) {

num = num \* 10 + (\*ip - '0');

} else if (\*ip == '.') {

if (num < 0 || num > 255) {

return 0;

}

num = 0;

dots++;

} else {

return 0;

}

ip++;

}

return (dots == 3 && num >= 0 && num <= 255);

}

// Function to validate port number

int is\_valid\_port(unsigned short port) {

return (port >= 1 && port <= 65535);

}

1. 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

ans:

#include <stdio.h>

#include <stdarg.h>

void log(const char \*format, ...) {

va\_list args; // Declare a va\_list to store the variable arguments

va\_start(args, format); // Initialize the va\_list with the format string

vfprintf(stdout, format, args); // Output the formatted string to stdout

va\_end(args); // Clean up the va\_list after use

}

int main() {

int count = 10;

char prefix = 'h';

char label[] = "India";

// Calling log() with format specifiers for integer, character, and string

log("count:%d, prefix:%c, label:%s\n", count, prefix, label);

return 0;

}

1. 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);

**ans:**

#include <stdio.h>

#include <stdarg.h>

#include <string.h>

char\* max\_len\_string(int num, ...) {

va\_list args;

va\_start(args, num);

char \*max\_str = va\_arg(args, char\*);

int max\_len = strlen(max\_str);

for (int i = 1; i < num; i++) {

char \*current\_str = va\_arg(args, char\*);

int current\_len = strlen(current\_str);

if (current\_len > max\_len) {

max\_str = current\_str;

max\_len = current\_len;

}

}

va\_end(args);

return max\_str;

}

int main() {

char \*ptr = max\_len\_string(5, "hi", "hello", "How", " Are", "END");

printf("%s\n", ptr);

return 0;

}