# Dynamic Memory Management Assignment

1. Write a program to read a line of text containing 2 or more words, tokenize, display the words, concatenate the words using ‘\_’ and display the final string. Consider line length of 80 characters. Provide a modular solution implementing following functions.

//process the input string and return a concatenated string allocated memory in heap

char \*process\_string(char \*line);

In main(), free the allocated memory after displaying the concatenated string

1. Code:



A screen shot of a computer program

Description automatically generated

Output:

A black background with white text

Description automatically generated

1. WAP to read a URL as input from the user, extract the host name and domain name, store them collectively in an appropriate data structure allocating dynamic memory for its members as per required length. Display the structure contents. Free the memory finally. Some of the functions to be implemented are:

//validate the received url

int isValidURL(char \*url);

//extract and return host name allocated memory in heap

char \*gethost(char \*url);

//extract and return domain name allocated memory in heap

char \*getdomain(char \*url);

void display(struct url \*obj);

void free(struct url obj);

Input: <http://www.altran.com>

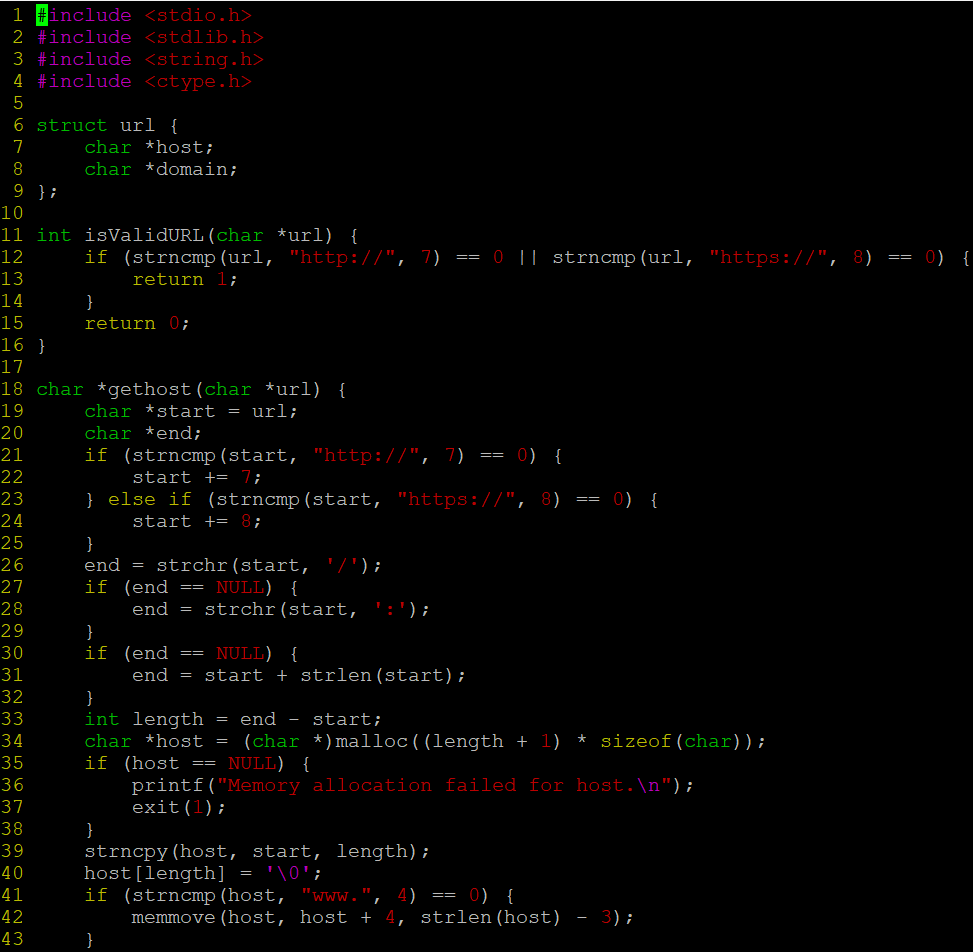
Output:

Host: altran

Domain: com

Specify the dataset used to test the program

1. Code:



A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Output:

A black background with white text

Description automatically generated

1. WAP to read a maximum of N (N is user input) strings or less from the user at runtime, each string could be of variable length not exceeding a maximum length of 80 characters, allocate memory in heap as per string length and store the strings. Stop reading inputs if input string is “end” or “END”. Display the stored strings. Free the memory before exiting program. Some of the functions to be implemented are:

[Note : Expected to use char \*\* and not fixed 2D array]

//allocate memory for a double pointer to hold n pointers and return the pointer

char \*\*allocate\_array\_memory(char \*\*ptr, int n);

//allocate memory for input string and return the pointer

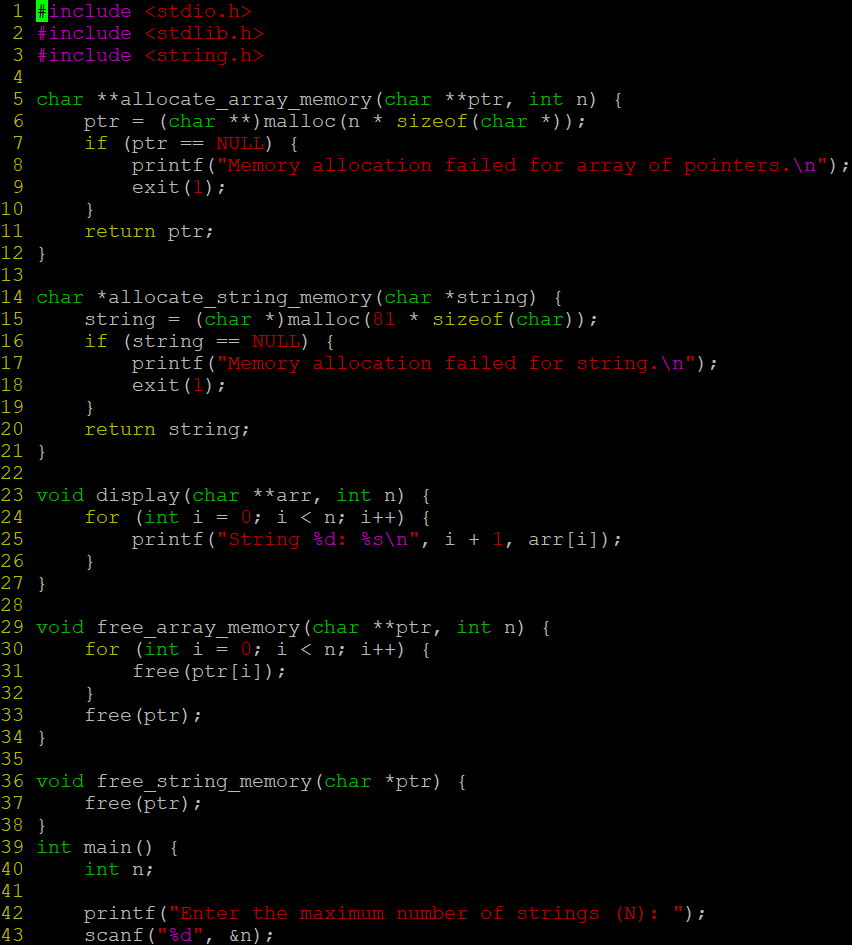
char \*allocate\_string\_memory(char \*string);

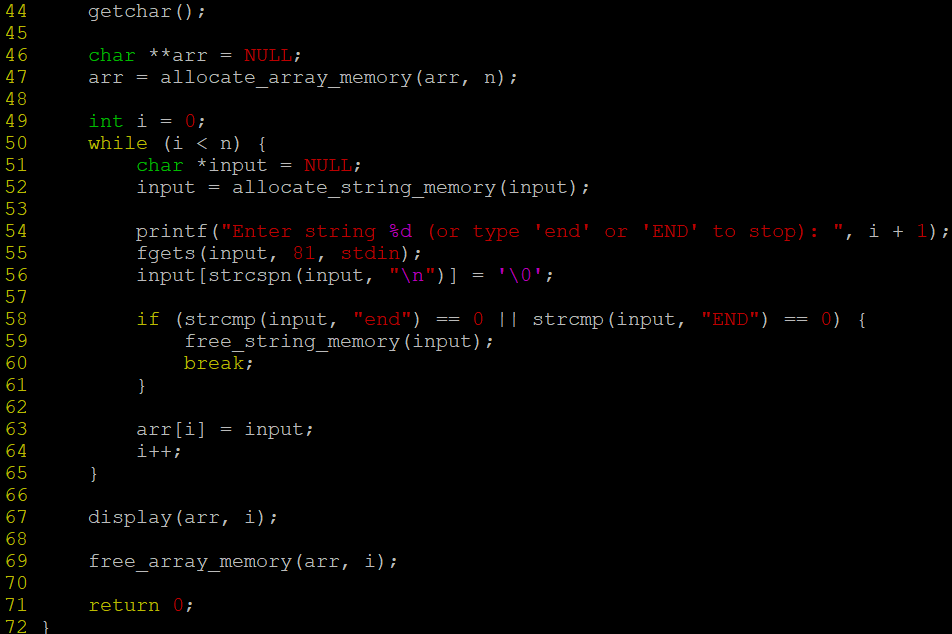
void display(char \*\*arr, int n);

void free\_array\_memory(char \*\*ptr, int n);

void free\_string\_memory(char \*ptr);

1. Code:





Output:

A screen shot of a computer

Description automatically generated