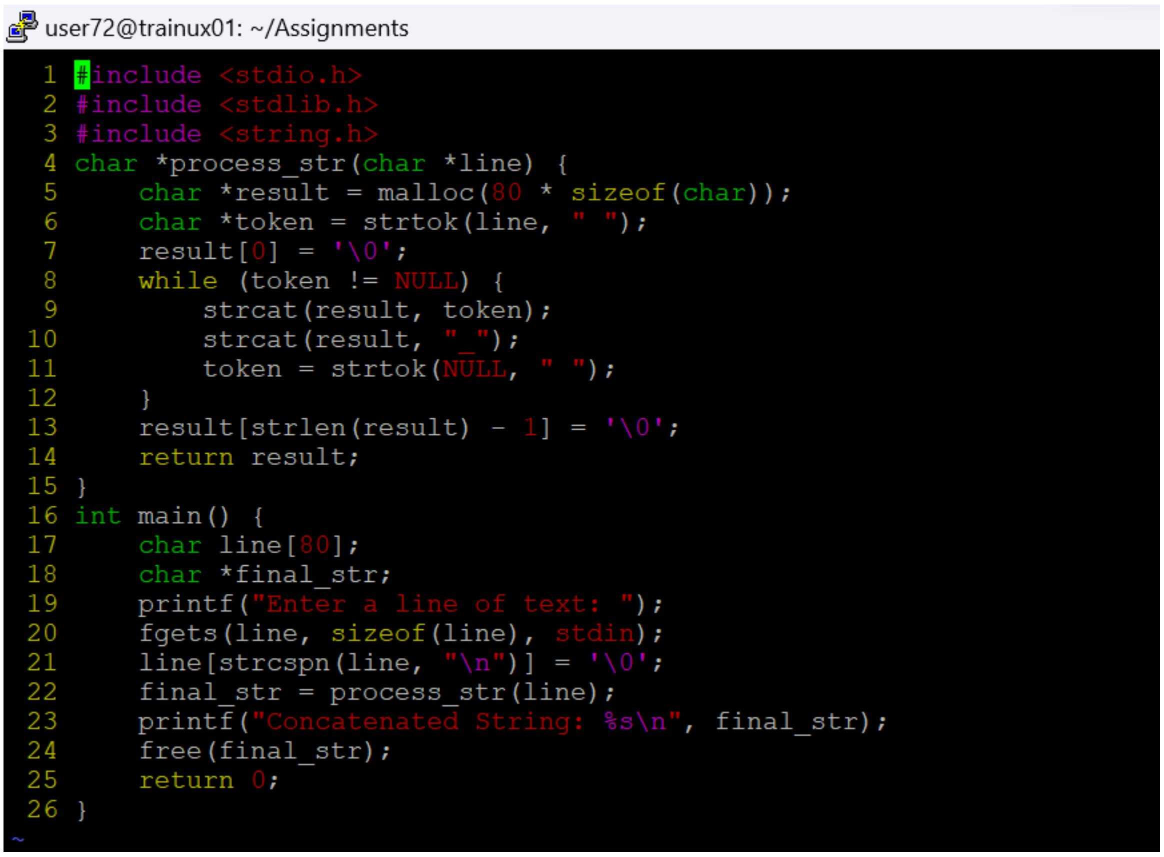
Dynamic Memory Management Assignment

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

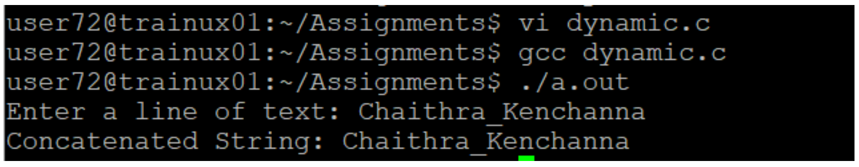
    ```
    char *process_string(char *line);
    ```

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



user72@trainux01: ~/Assignments

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char *process_str(char *line) {
    char *result = malloc(80 * sizeof(char));
    char *token = strtok(line, " ");
    result[0] = '\0';
    while (token != NULL) {
        strcat(result, token);
        strcat(result, "_");
        token = strtok(NULL, " ");
    }
    result[strlen(result) - 1] = '\0';
    return result;
}
int main() {
    char line[80];
    char *final_str;
    printf("Enter a line of text: ");
    fgets(line, sizeof(line), stdin);
    line[strcspn(line, "\n")] = '\0';
    final_str = process_str(line);
    printf("Concatenated String: %s\n", final_str);
    free(final_str);
    return 0;
}
~
```



```
user72@trainux01:~/Assignments$ vi dynamic.c
user72@trainux01:~/Assignments$ gcc dynamic.c
user72@trainux01:~/Assignments$ ./a.out
Enter a line of text: Chaithra_Kenchanna
Concatenated String: Chaithra_Kenchanna
```

2. 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

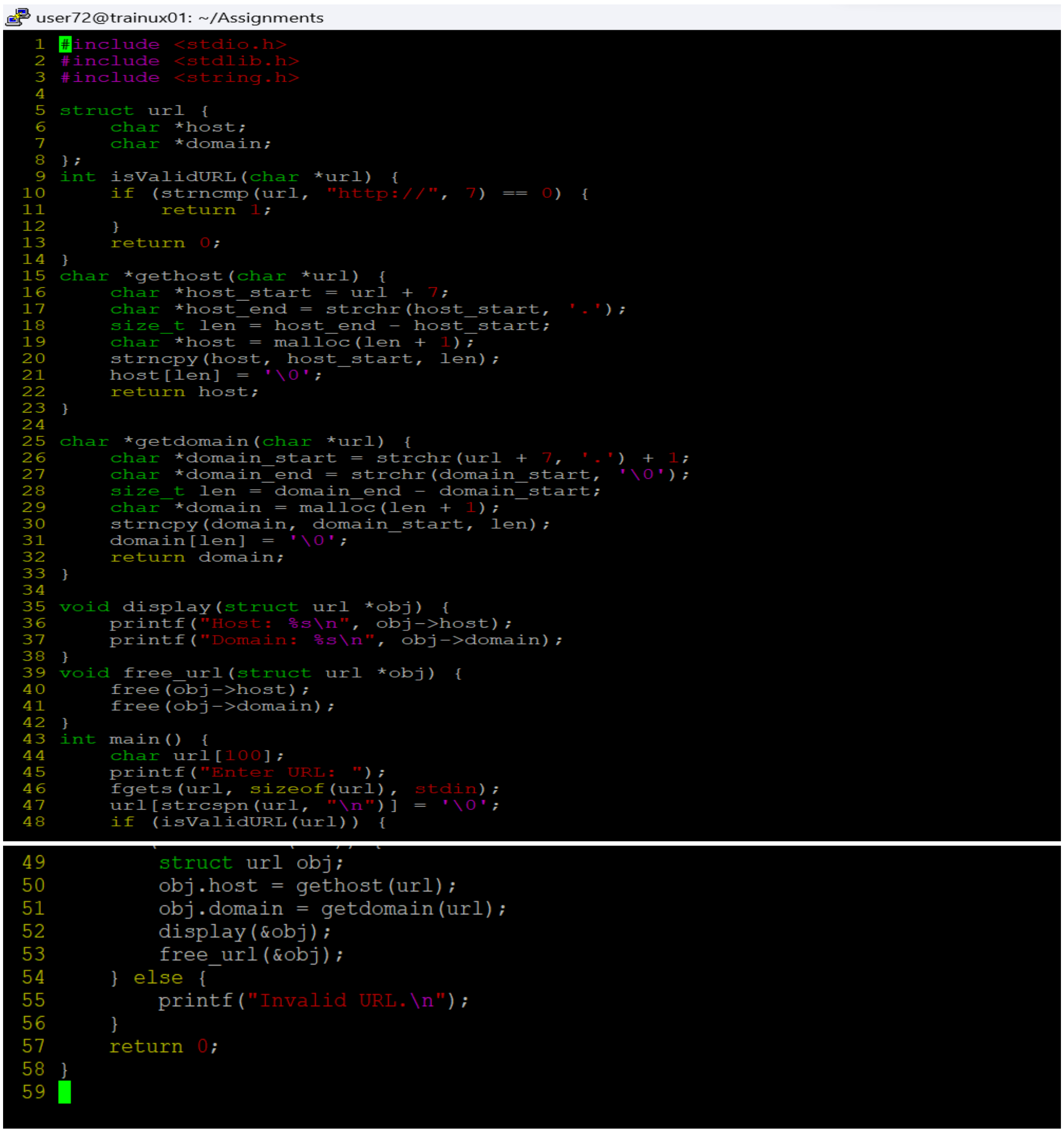
    ```
    int isValidURL(char *url);
    ```

```
//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:

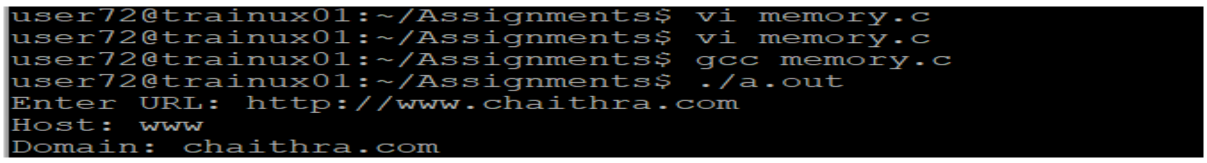
```
Host: altran
Domain: com
```

Specify the dataset used to test the program



user72@trainux01: ~/Assignments

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct url {
    char *host;
    char *domain;
};
int isValidURL(char *url) {
    if (strncmp(url, "http://", 7) == 0) {
        return 1;
    }
    return 0;
}
char *gethost(char *url) {
    char *host_start = url + 7;
    char *host_end = strchr(host_start, '.');
    size_t len = host_end - host_start;
    char *host = malloc(len + 1);
    strncpy(host, host_start, len);
    host[len] = '\0';
    return host;
}

char *getdomain(char *url) {
    char *domain_start = strchr(url + 7, '.') + 1;
    char *domain_end = strchr(domain_start, '\0');
    size_t len = domain_end - domain_start;
    char *domain = malloc(len + 1);
    strncpy(domain, domain_start, len);
    domain[len] = '\0';
    return domain;
}

void display(struct url *obj) {
    printf("Host: %s\n", obj->host);
    printf("Domain: %s\n", obj->domain);
}
void free_url(struct url *obj) {
    free(obj->host);
    free(obj->domain);
}
int main() {
    char url[100];
    printf("Enter URL: ");
    fgets(url, sizeof(url), stdin);
    url[strcspn(url, "\n")] = '\0';
    if (isValidURL(url)) {
        struct url obj;
        obj.host = gethost(url);
        obj.domain = getdomain(url);
        display(&obj);
        free_url(&obj);
    } else {
        printf("Invalid URL.\n");
    }
    return 0;
}

```

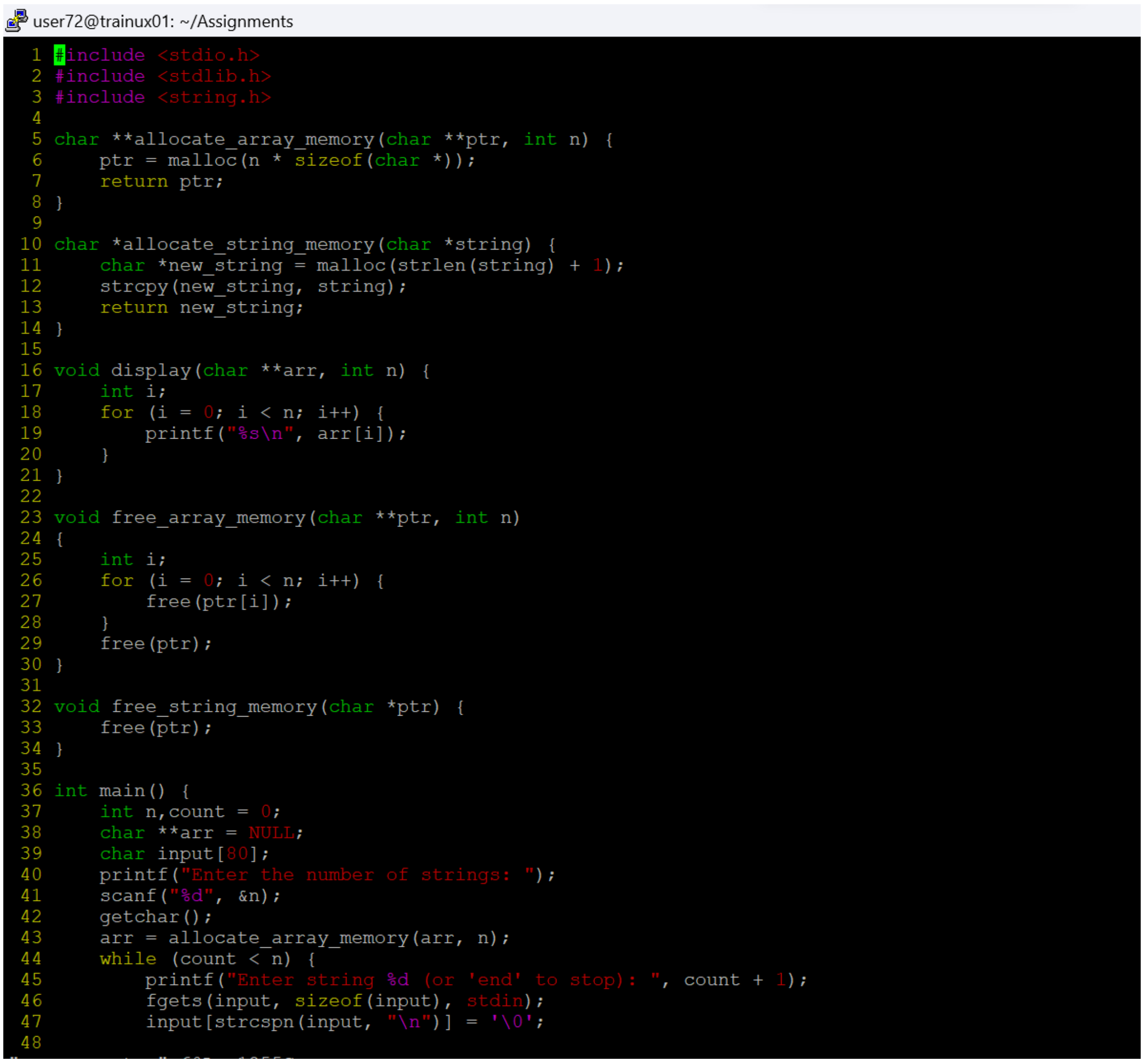


```
user72@trainux01:~/Assignments$ vi memory.c
user72@trainux01:~/Assignments$ vi memory.c
user72@trainux01:~/Assignments$ gcc memory.c
user72@trainux01:~/Assignments$ ./a.out
Enter URL: http://www.chaithra.com
Host: www
Domain: chaithra.com
```

3. 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:

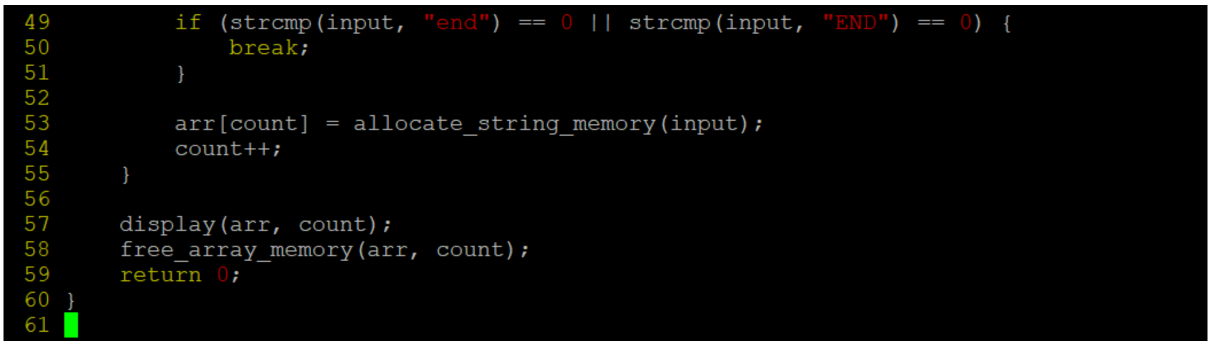
[Note : Expected to use char ** and not fixed 2D array]

```
//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);
```

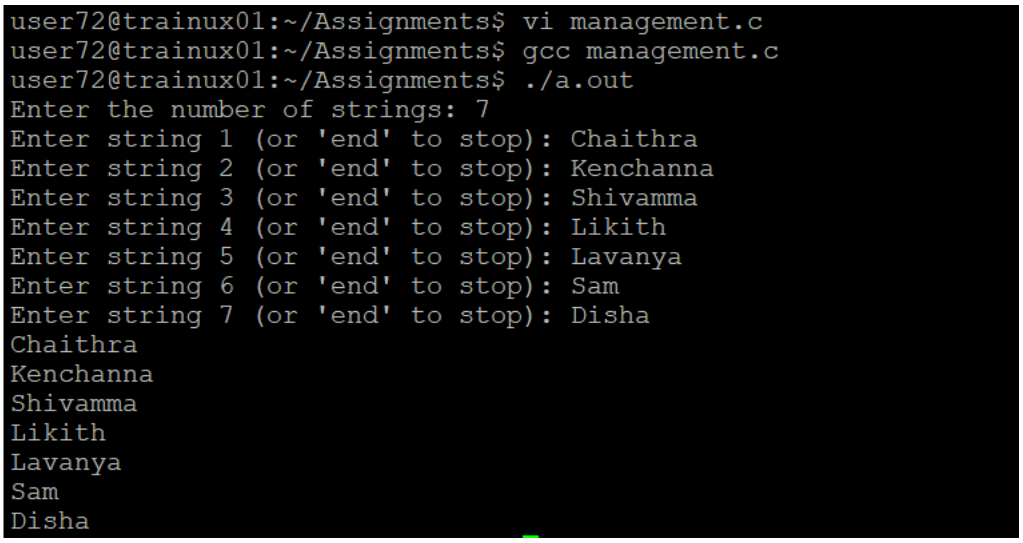


user72@trainux01: ~/Assignments

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char **allocate_array_memory(char **ptr, int n) {
    ptr = malloc(n * sizeof(char *));
    return ptr;
}

char *allocate_string_memory(char *string) {
    char *new_string = malloc(strlen(string) + 1);
    strcpy(new_string, string);
    return new_string;
}

void display(char **arr, int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%s\n", arr[i]);
    }
}

void free_array_memory(char **ptr, int n)
{
    int i;
    for (i = 0; i < n; i++) {
        free(ptr[i]);
    }
    free(ptr);
}

void free_string_memory(char *ptr) {
    free(ptr);
}

int main() {
    int n,count = 0;
    char **arr = NULL;
    char input[80];
    printf("Enter the number of strings: ");
    scanf("%d", &n);
    getchar();
    arr = allocate_array_memory(arr, n);
    while (count < n) {
        printf("Enter string %d (or 'end' to stop): ", count + 1);
        fgets(input, sizeof(input), stdin);
        input[strcspn(input, "\n")] = '\0';

```



```
        if (strcmp(input, "end") == 0 || strcmp(input, "END") == 0) {
            break;
        }

        arr[count] = allocate_string_memory(input);
        count++;
    }

    display(arr, count);
    free_array_memory(arr, count);
    return 0;
}

```



```
user72@trainux01:~/Assignments$ vi management.c
user72@trainux01:~/Assignments$ gcc management.c
user72@trainux01:~/Assignments$ ./a.out
Enter the number of strings: 7
Enter string 1 (or 'end' to stop): Chaithra
Enter string 2 (or 'end' to stop): Kenchanna
Enter string 3 (or 'end' to stop): Shivamma
Enter string 4 (or 'end' to stop): Likith
Enter string 5 (or 'end' to stop): Lavanya
Enter string 6 (or 'end' to stop): Sam
Enter string 7 (or 'end' to stop): Disha
Chaithra
Kenchanna
Shivamma
Likith
Lavanya
Sam
Disha
```