Learning summary:
Study: 5.5 Hours
Exercises: 2.5 Hours

Documentation of Day_21

Exercise 1-16:

Revise the main routine of the longest-line program so it will correctly print the length of arbitrary long input lines, and as much as possible of the text.

Source Code:

```
#include <stdio.h>
#define MAXLINE 10 /* maximum input line length */
int getline(char line[], int maxline);
void copy(char to[], char from[]);

int main() {
int len; /* current line length */
int max; /* maximum length seen so far */
char line[MAXLINE]; /* current input line */
char longest[MAXLINE]; /* longest line saved here */
max = 0;
        while ((len = getline(line, MAXLINE)) > 0)
        if (len > max) {
                max = len;
                copy(longest, line);
        }
        if (max > 0) /* there was a line */
                printf("%s", longest);
return 0;
}

int getline(char s[],int lim) {
int c, i;
        for (i=0; i < lim-1 && (c=getchar())!=EOF && c!='\n'; ++i)
                s[i] = c;
        if (c == '\n') {
                s[i] = c;
                ++i;
        }
s[i] = '\0';
        while(c != EOF && c != '\n') {
   ++i;
   c = getchar();
        }
return i;
}
```

```
void copy(char to[], char from[]) {
int i;
i = 0;
        while ((to[i] = from[i]) != '\0')
                ++i;
}
```

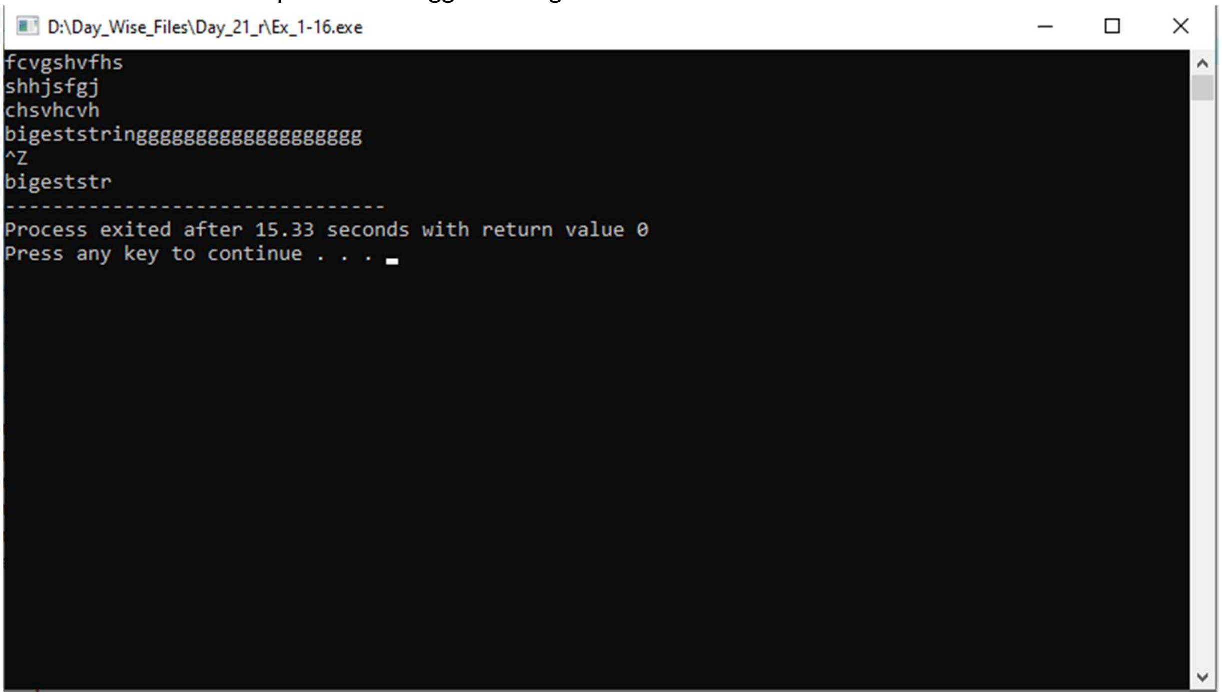
I have implemented this program that reads input lines and prints the longest line among them.

The program uses the getline function to read a line from input and store it in an array. The function copy is used to copy the contents of one array to another.

In the main function, I have initialized variables len and max to keep track of the current line length and the maximum length seen so far, respectively. The function repeatedly calls getline until the end of file is reached. If the length of the line is greater than the previous maximum length, it updates the maximum length and copies the line to the longest array. Finally, if there was a line, it prints the longest line.

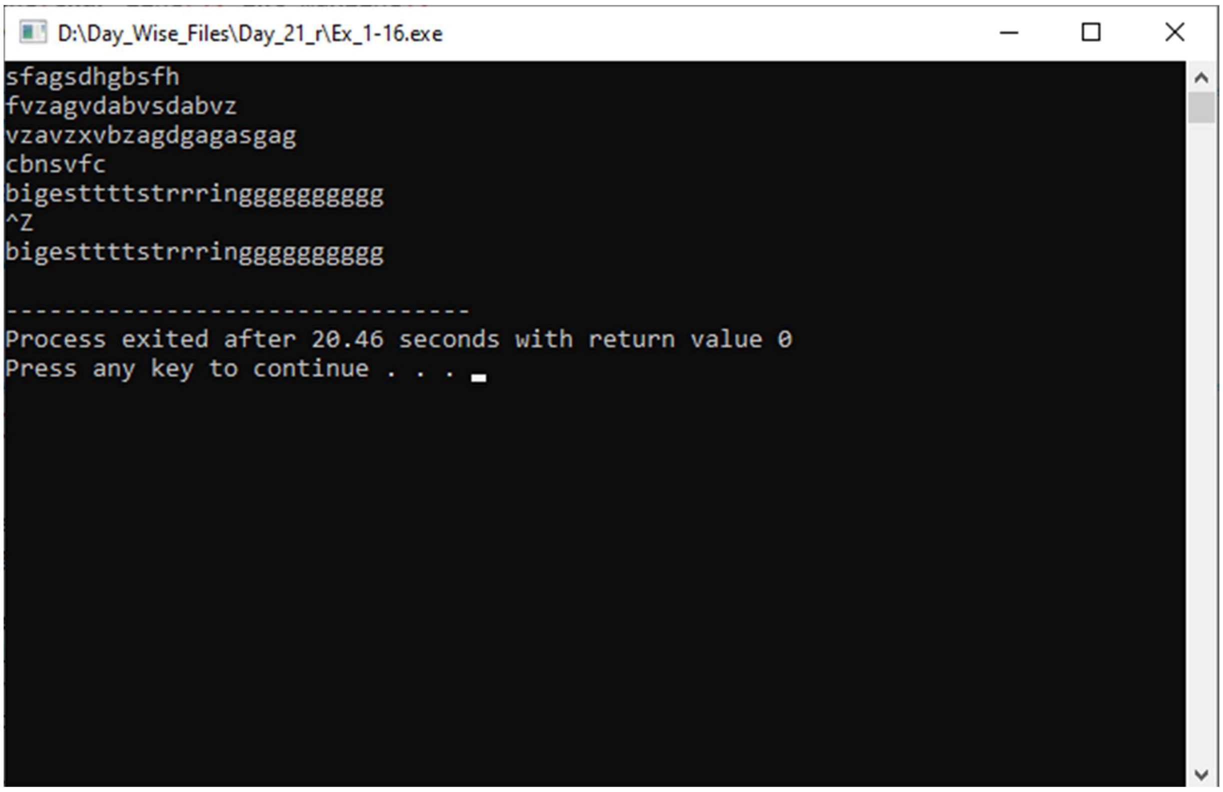
Inputs and Outputs:

1. In this input output scenario the maximum line length is set to 10 but if we give more than 10 sized string the program will also count the maximum length of it and show the first 10th characters as a part of the biggest string.



```
fcvgshvfhs
shhjsfgj
chsvhcvh
bigeststringggggggggggggggggggg
^Z
bigeststr
--------------------------------
Process exited after 15.33 seconds with return value 0
Press any key to continue . . .
```

2. This time the maximum string length is set to 100 so that the biggest string is printed here as it size is less than 100.



D:\Day_Wise_Files\Day_21_r\Ex_1-16.exe

```
sfagsdhgbsfh
fvzagvdabvsdabvz
vzavzxvbzagdgagasgag
cbnsvfc
bigesttttstrrringgggggggggg
^Z
bigesttttstrrringgggggggggg

--------------------------------
Process exited after 20.46 seconds with return value 0
Press any key to continue . . .
```

Exercise 1-18:

Write a program to remove trailing blanks and tabs from each line of input, and to delete entirely blank lines.

Source Code:

```
#include <stdio.h>
#include <string.h>
#define MAXLINE 1000 /* maximum input line length */
int getline(char line[], int maxline);
void copy(char to[], char from[]);

int main() {
int len; /* current line length */
char line[MAXLINE]; /* current input line */
char longest[MAXLINE]; /* longest line saved here */
while ((len = getline(line, MAXLINE)) > 0) {
                int n;
    n = strlen(line);
    printf("Length before omitting spaces is : %d\n", n-1);
    for (n = strlen(line)-1; n >= 0; n--)
                        if (line[n] != ' ' && line[n] != '\t' && line[n] != '\n')
                        break;
                line[n+1] = '\0';
    if (line[0] == '\n')
    ;
                n = strlen(line);
    printf("Length after omitting spaces is : %d\n", n);
  }
  return 0;
}

int getline(char s[],int lim) {
        int c, i;
        for (i=0; i < lim-1 && (c=getchar())!=EOF && c!='\n'; ++i)
                s[i] = c;
        if (c == '\n') {
                s[i] = c;
                ++i;
        }
        s[i] = '\0';
return i;
}
void copy(char to[], char from[]) {
        int i;
        i = 0;
        while ((to[i] = from[i]) != '\0')
                ++i;
}
```

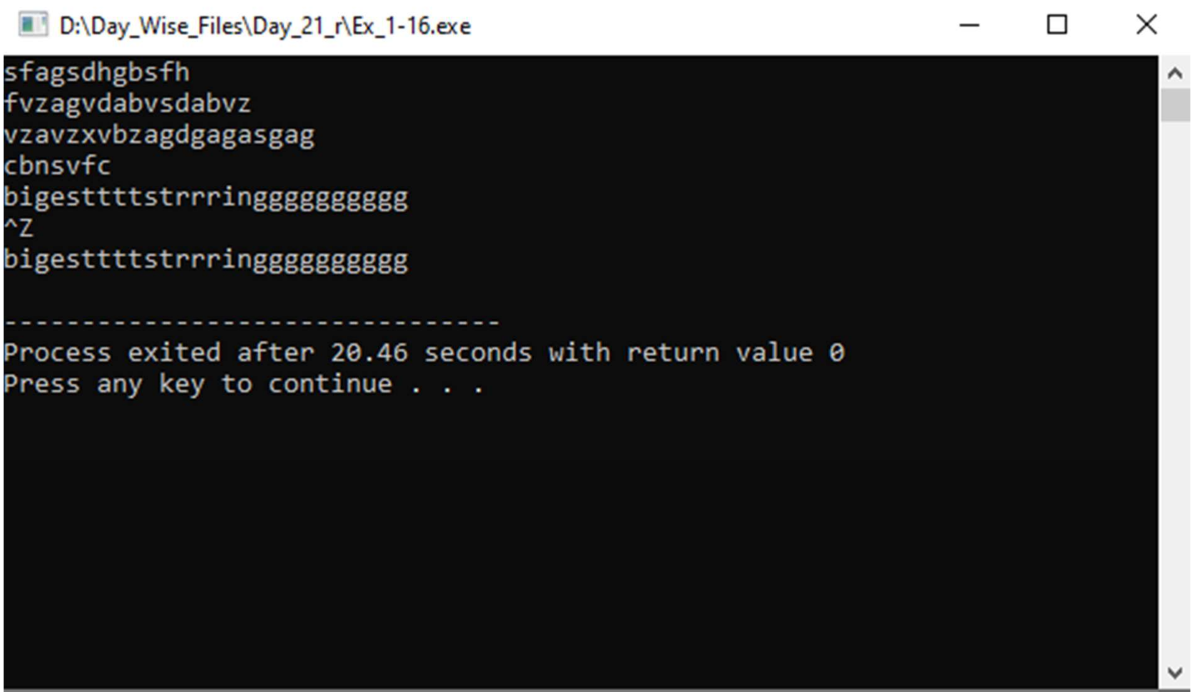
The program that I have implemented an extension of the program of the previous exercise. It reads input lines, omits trailing spaces and tabs, and then prints the length of each line before and after removing these characters. It uses the getline function to read a line of input and stores it in an array.

In the main function, I have initialized variables len, n, line, and longest. It repeatedly calls getline until the end of file or a newline character is encountered. It calculates the length of the line before omitting spaces and tabs, and then it removes any trailing spaces and tabs by replacing them with a null character. It checks if the line is empty (only containing a newline character) and skips further processing. Finally, it calculates the length of the line after omitting spaces and tabs and prints both lengths.

The getline function reads characters from input until it reaches the maximum line length (lim), the end of file, or a newline character. It stores the characters in the s array and appends a null character at the end to signify the end of the string. It also counts the total number of characters read and ignores any extra characters beyond the specified limit.

The copy function simply copies the characters from the from[] array to the to[] array until it reaches the null('\0') character, effectively creating a duplicate of the string.

Input and Output:



D:\Day_Wise_Files\Day_21_r\Ex_1-16.exe

```
sfagsdhgbsfh
fvzagvdabvsdabvz
vzavzxvbzagdgagasgag
cbnsvfc
bigesttttstrrringgggggggg
^Z
bigesttttstrrringgggggggg

--------------------------------
Process exited after 20.46 seconds with return value 0
Press any key to continue . . .
```