CSE 115L: Programming Language I Lab (Section: 06)

Spring 2020

Final Assignment

Marks : 80

Name:	**Ihsanul Haque Asif**
ID:	**2013664642**

Tasks

1. Write a C program that prompts the user to enter a string that contains both alphabets and digits and then find the frequency of each digit.
   You should print the frequency of each digit (0 to 9) in 10 space separated integers.
   In the following example, the string contains-
   - 0- zero time, 1- two time, 2- three times, 3- three times,
   - 4, 5, 6, 7- zero time each, 8 and 9- one time each

   (25 points)

Sample input	Sample output
Enter a string: Good 239 f923d 118luck 32	Digit occurrence: 0 2 3 3 0 0 0 0 1 1

   **you can use string functions

Code	Description

#include<stdio.h>
 	
int main(void)
{
printf("Enter a string:\n");
 	
char string[1000];
gets(string);
 	
printf("\nDigit occurrence:\n");
int digit_count = 0;
 	
for(int i=0;i<=9;i++)
{
for(int j=0;string[j]!='\0';j++)
{
if(string[j]-48 == i)
digit_count++;
}
 	
printf("%d ",digit_count);
digit_count = 0;
}
 	
return 0;
}
	1. At first print a message for the user to enter a string.
	2. Declared a char type array name “string”, I define its size 1000.
	3. Get the string from user input using gets() function and store it in “string”.
	
	4. Print another message for the user.
	5. Declare an integer type variable named “digit_count” and assign 0 to it.
	
	6. Now I used nested “for loop”. In the 1st loop declare a loop control variable “i” and assign 0 into the loop already. The condition is i<=9. That means this loop will run 0 to 9. And change the value by “i++”.
	7. In the 2nd loop, the loop control variable declared “j” and assigned 0. And the condition is string[j]!='\0’. That means this loop will run until strings “ith” index is not NULL. index moved by “j+”.
	8. Under the nested loop here I implement a “if condition” to check whether the difference between string’s “jth” character’s ASCII value and ZERO’s ASCII value(48) is equals to “i”.
	Ex: 57-48 = 9. Here 57 is the ASCII value of 9.
	
	If this condition is true then the “digit_count” variable will increment 1 (digit_count++). And print the frequency of the digit under the 1st loop.
	
	After print once, again set the value of “digit_count” to 0.
	
	This operation will continue until the nested loop is satisfied.
	
	And here my program will finish all the operations and **STOP**.

2. Write a C program that takes a string as input and finds the length of that string. You need to solve the problem using pointer and NOT string function.

(15 points)

Sample input	Sample output
Enter a string: Good luck!	Length of the string: 10

Code	**Description**

#include<stdio.h>
 	
int main(void)
{
printf("Enter a string:\n");
char string[1000];
gets(string);
 	
char *ptr;
ptr = string;
 	
int count = 0;
for(int i=0;*(ptr+i)!='\0';i++)
count++;
 	
printf("\nLength of the string: %d",count);
 	
return 0;
}
	1. At first I print a message for the user to enter a string then a new line.
	2. Declared a char type array named "string" and fixed the size 1000.
	3. Get a string as user input and store it in the "string" variable.
	
	4. Declare a char type pointer, "*ptr" variable.
	5. Assign the "string" to "ptr". Now this pointer can use the "string" and it's address.
	
	6. Declare an integer type variable named "count" and assign 0 to it. This will be used to count the length of "string".
	
	7. Implement a "for loop"
	Declare a loop control variable "i" assign 0 to it. This "i" will control the loop. And the condition for this loop is *(ptr+i)!='\0'.
	
	Now i'm describing *(ptr+i)!='\0' :
	
	a. *ptr indicates the 1st element of the "string" (0th index).
	
	b. Then *(ptr+i) will move the array index (via loop control variable i)
	
	c. '\0' means NULL. This operation will continue until ith index is no NULL.

	8. Under this loop the “count” variable will be incrementing (counting length) until the loop is satisfied. 9. After finishing counting I printed the value of the count variable. 10. The value of the count variable is the length of the “string” . And here my program will finish all the operations and **STOP**.

3. Write a C program that takes a number as input and then finds the sum of the digits using recursion. [Hint: Use the remainder and division operators].

(15 points)

Code	**Description**

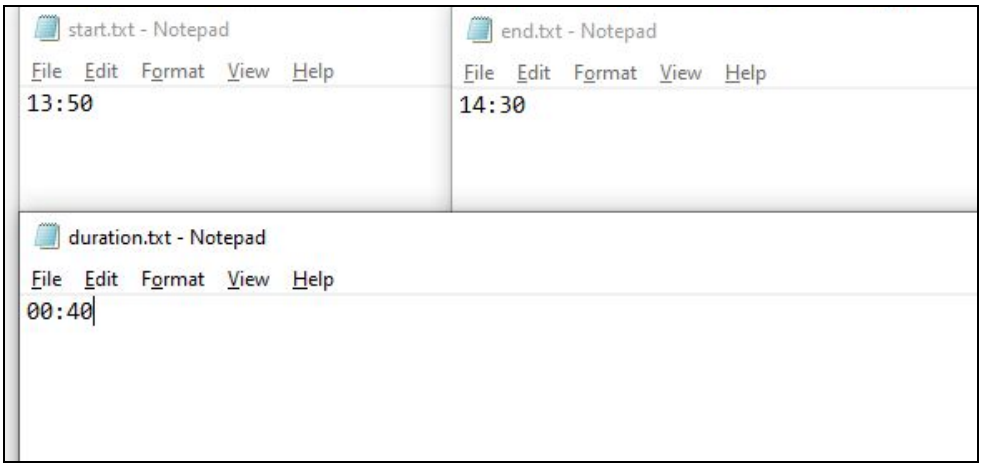
#include<stdio.h>
 	
int sum_of_digits(int n)
{
  if(n == 0)
  return 0;
  else
  return ((n%10)+sum_of_digits(n/10));
}
 	
int main(void)
{
  int num;
  printf("Enter digits: ");
  scanf("%d",&num);
 	
  int sum = sum_of_digits(num);
  printf("\nSum of digits: %d",sum);
 	
  return 0;
}
	main() function :
	
	1. Declare an integer type variable (num).
	2. print a message "Enter digits:".
	3. getting input an integer number from the user and store it in “num” variable
	
	4. Again declare an integer type variable (sum) and assign it to the function called "sum_of_digits", and pass the "num" variable through it.
	
	5. As I called my recursive function now other tasks will be done by the function and return values at the “sum” variable and keep adding until the number becomes 0. At last print the sum of digits here in main() function.
	
	sum_of_digits() function :
	
	This function gets an integer value (get from the user) from the main function, and I declared it as a new variable "n" (integer type).
	
	1. Firstly here I write a conditional statement to check if "n" is equal to 0 or not.
	
	2. If "n" is equal to 0 then this function will return 0 to the main function (this will store in the sum variable). This is the base case.
	
	3. If "n" is not equal to 0 then this function will return ((n%10)+sum_of_digits(n/10)). This is the recursive case.

	Here sum_of_digits (n/10) means this function calls itself again and again until the value of “n” becomes 0. **return ((n%10)+sum_of_digits(n/10))** **Described here :** Suppose we get "num" from user input. So the main function will pass "num" to the function called "sum_of_digits". After getting "num", the conditional statement will check whether it's 0 or not. If it's not 0 then the else statement will work with it. 1. Firstly, find the last digit of the number using modular division by 10. 2. Secondly, add the last digit (which we found by step 1) to the "sum" variable. 3. Thirdly, remove the last digit from the number by dividing it by 10 and then pass this again to the function called "sum_of_digit" (Calling itself). This is the recursive approach sum(num) = (n%10)+sum_of_digits(n/10). This function will repeat these three steps till the number becomes 0. And then print the sum of the digits. And here my program will finish all the operations and **STOP**.

4. Write a program that calculates the duration of a class given the starting and ending times and write the result in a file “duration.txt”.

   - ➢ Create a structure- “Time” with the following components and appropriate data types: hour & minute.
   - ➢ The starting time of the class should be read from a file “start.txt” and the end time should be read from another file “end.txt”.
   - ➢ Time format in the files must be hh:mm.

(25 points)



Code	**Description**

#include <stdio.h>
struct Time
{
  int hour;
  int minute;
};
 	
int main()
{
  struct Time start,stop,res;
  char ch;
 	
  //start.txt
  FILE *fptr1;
  fptr1 = fopen("start.txt","r");
	1. Before I write my code I created two .txt files in the same folder where my program will run. One file is **start.txt** that contains the start time and another is **end.txt** file which contains the end time.
	
	Defined a **structure** is named “Time”. Under this there are two integer type variables “hour” and “minute”.
	
	2. **main()** function starts from here :
	
	Create three structure variables named “start”, “stop” and “res”. Now we can work with it.

```
  if(fptr1 == NULL)
     printf("\nFILE NOT FOUND!\n");


fscanf(fptr1,"%d%c%d",&start.hour,&ch,&start.minute)
;
  fclose(fptr1);


  //end.txt
  FILE *fptr2;
  fptr2 = fopen("end.txt","r");

  if(fptr2 == NULL)
     printf("\nFILE NOT FOUND!\n");


fscanf(fptr2,"%d%c%d",&stop.hour,&ch,&stop.minute);
  fclose(fptr2);


   //convert in minute
   start.hour*=60;
   stop.hour*=60;

   start.hour+=start.minute;
   stop.hour+=stop.minute;

   //subtracting
   int temp = abs(start.hour-stop.hour);


  //duration.txt
  FILE *fptr3;
  fptr3 = fopen("duration.txt","w");

  if(fptr3 == NULL)
     printf("\nFILE NOT FOUND!\n");


  if(temp < 60)
  {
     if(temp < 10)
      fprintf(fptr3,"00:0%d",temp);
     else
      fprintf(fptr3,"00:%d",temp);
  }
```

3. Declare a char type variable “ch” to store this character “ : ” from file.

4. Now I declare a pointer type FILE named “fptr1” (* is for pointer notation). Then open a file using fopen(). This fopen() is assigned in fptr1. This file will open in reading mood so that I use “r” here. And “start.txt” is that file which will be opened for reading data.

5. Set a condition to check if the file exists or not. If not then my program will print a message “FILE NOT FOUND!”.

6. As we opened the start.txt file for reading now my program will read an integer then a character then another integer (13:50 saved before) from this file using fscanf().

7. In the fscanf() function we can read the file and store data into my structure variables. Here I store the two integer numbers into structure variables “start.hour” and “start.minute”. Following time format (hh:mm).

8. Now I have the start hour and minute stored in structure variables. So I close the opened FILE using fclose().

9. My program will also read the end time from the file “end.txt” following the same process in point 4 to 8.

   Difference is here the FILE pointer variable name is “fptr2” and file name is “end.txt” (saved before). Program will read two int type data and store them in my other two structure variables “stop.hour” and “stop.minute” and the character “ : ” stored in “ch” variable (for ignore).

   Then again close this fptr2 FILE too.

```
  else if(temp >= 60)
  {
     res.hour = temp/60;
     res.minute = temp%60;

     if(res.hour < 10)
     {
        if(res.minute < 10)
         fprintf(fptr3,"0%d:0%d",res.hour,res.minute);
        else if(res.minute >= 10)
         fprintf(fptr3,"0%d:%d",res.hour,res.minute);
     }
     else
      fprintf(fptr3,"%d:0%d",res.hour,res.minute);

  }

  fclose(fptr3);
  return 0;
}
```

10. So we have start and end time now my task is to calculate the difference between them and write it in a file.

11. Firstly converted starting and ending hour into minute and then added existing minutes. Following those steps below:

    Step 1: start.hour*=60;
    (start.hour = start.hour*60)
    Here I multiply the start hour by 60 and convert it into minutes.

    Ex: start.hour = 13*60 = 780 minutes

    Step 2: stop.hour*=60;
    (stop.hour = stop.hour*60)
    Here I multiply the end hour by 60 and convert it into minutes.

    Ex: stop.hour = 14*60 = 840 minutes

    Step 3: start.hour+=start.minute;
    (start.hour = start.hour+start.minute)
    Here I added the extra minutes with strat.hour (converted into minute) and got a total time in minutes.

    Ex: start.hour = 780+50 = 830 minutes

    Step 4: stop.hour+=stop.minute;
    (stop.hour = stop.hour+stop.minute)
    Here I added the extra minutes with stop.hour (converted into minute) and got a total time in minutes.

    Ex: stop.hour = 840+30 = 870 minutes

12. Now I have start and end time in minutes. So I can easily get the difference between them by subtracting them.

    Declare an int type variable “temp” to store the difference.

    Subtract the end time from start time using abs() function. This abs() function will give me the absolute value. Because after subtracting the value could be negative but time can’t be negative so I use abs() here. And then assign the value in “temp”. Now “temp” is storing the difference.

    For better understanding :
    start.hour = 830
    stop.hour = 870
    start.hour-stop.hour = 830-870 = -40

    temp = abs(start.hour-stop.hour);
    temp = abs(-40)
    temp = 40

13. Again declared a pointer type FILE named “fptr3” (* is for pointer notation).

    Then open a file using fopen(). This fopen() is assigned in fptr3. This file will open in writing mood so that I use “w” here. And **“duration.txt”** is that file which will be created for writing data.

14. Set a condition to check if the file exists or not. If not then my program will print a message “FILE NOT FOUND!”. Else keep doing tasks.

15. Now I implement a nested if-else condition to check whether the difference between start and end time (stored in “temp”) is less than 60 (minutes) or not.

    if the time difference (temp < 60) is less than 60 (minutes) then again

    if condition will check if the duration is less than 10 (minutes) or not (temp < 10). If the duration is less than 10 then my program will print 00:0 and then the time in minutes in “duration.txt” FILE using fprintf().

    else my program will print 00: then the time in minutes in “duration.txt” FILE using fprintf(). fprintf() function is used to write something in file.

    else if the time difference is greater or equal (temp >= 60) than 60 (minutes) that means the time difference can be one hour or greater. Now my program will divide the minutes into hours in hh:mm format. Following steps below :

    Step 1: res.hour = temp/60;

    Here “res” was the structure variable. This is used for store difference hours.

    “temp” is divisioned by 60 and the quotient is assigned to res.hour. The quotient is the duration hour here.

    Step 2: res.minute = temp%60;

    Here simply the reminder of the division assigned into res.minute. The reminder is the duration minute here.

For better understanding :
suppose “temp” = 115 (minutes)

res.hour = 115/60 = 1
res.minute = 115%60 = 55
So, 1:55 (hh:mm) is the difference.

16. Inside the **if-else** condition here I implement again a nested **if-else** for format the output in FILE.

    if (res.hour < 10) res.hour (duration hour) is less than 10. Inside this if statement implement another nested loop for formatting the print in FILE.

        if (res.minute < 10) res.minute (duration minute) is less than 10, then the printing format will be “0%d:0%d”.

        else if (res.minute >= 10) res.minute (duration minute) is greater than 10 or equal, then the printing format will be “0%d:%d”.

    else
    The format will be “%d:0%d”.

After successfully printing the time duration into the “duration.txt” FILE this fptr3 FILE will be closed using fclose().

And here my program will finish all the tasks and **STOP**.