**ANALYZING A LINE OF ALPHANUMERIC TEXT**



**SUBMITTED BY:**

***Team name:* DIVISIBLE BY ZERO**

AISHWARYA B 16TUEE010

PRIYADARSHINI B 16TUEE149

VANATHI T 16EUIT165

october 2018

sri krishna INSTITUTIONS

(autonomous)

Coimbatore 641042

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO** |
| 1 | About us, Contact us | 3 |
| 2 | Abstract | 4 |
| 3 | Objective | 5 |
| 4 | Existing system | 6 |
| 5 | Proposed system | 6 |
| 6 | Specification | 6 |
| 7 | HDD | 7 |
| 8 | Sample screen shots | 9 |
| 9 | LDD | 11 |
| 10 | Control Flow Diagram | 13 |
| 11 | Pseudo code | 15 |
| 12 | Algorithm | 17 |
| 13 | Flow chart | 19 |
| 14 | Code | 25 |
| 15 | Quality Assurance - Test cases | 33 |
| 16 | Output screens | 34 |
| 17 | Complexity -No of lines and Execution time | 35 |
| 18 | Conclusion | 35 |
| 19 | References | 36 |

Guide signature

**1. ABOUT US:**

As unique as we are individually, we share a mutual passion for technology and doing excellent work for our clients. We pride ourselves on our integrated approach to public relations and the results it delivers. We strive for excellence in every day action.

* 1. **CONTACT US:**

Don’t be a stranger just say hello we are on reach. Accelerate your performance by staying current with the latest updates and taking features and functions developed to help you to be more productive.

**2. ABSTRACT:**

The major issue that people face most of the time is rules or specifications at instances such as registering for any applications. The people at the user level are unaware of the specifications of username and password. Thus our project helps in solving the issues with registration by avoiding confusions happening due to duplication and weak passwords. This system supports alphanumeric and special characters also. The Alphanumeric Text Analyzer gets a line of text as input and analyses each character and display it's specifications as per American Standard Code for Information Interchange. The application gets a username and checks for duplicates. If the user name entered is already taken, the user is asked to re-enter a new username else it is accepted. Standard specifications of a strong password includes presence of at least 8 characters with minimum 2 digits, one uppercase letter, one lowercase letter, one special character. If the password satisfies the specifications it is accepted by the application else the user is asked to re-enter a new password.

**3. OBJECTIVE:**

The prime objective of our project is to analyze alphanumeric text. Additionally our project evaluates the count of requested features of text. The project includes eccentric applications of alpha numerals such as the validation of username by not accepting the duplicates and by validating an alphanumeric password according to standard norms. Whenever we are new to any application, say it online or offline application, it asks for registration and here comes in the usage of validation of username and password. For instance, sign up process in Instagram (if you are a new user) can be quoted as an example.

**4. EXISTING SYSTEM:**

Alphanumeric text is not a new one. All registration includes alpha numerals. The password settings are of different types. Alpha numerals plays a vital role in day to day life.

**5. PROPOSED SYTEM:**

Our project is to carefully study text. Also our project the count of requested features of text. The project includes alpha numbers such as the validation of username by not accepting the copies and by validating a (with letters and numbers) password according to standard normal behaviors. Whenever we are new to any (online or paper form that asks for a job, money, admission, etc.), say it online or offline.

**6. SPECIFICATIONS:**

Module 1 -Get input and ASCII specifications

Module 2 - Count

Module 3 - User id creation (validity check)

Module 4 - Password generation module

Module 5 - Password validation module.

**7. HDD:**

**01. BACKGROUND**

This document contains the high level design of the project that analyzes a line of alphanumeric text by examining characters. The modules of the project are yet to be implemented.

|  |  |  |
| --- | --- | --- |
| **SNO** | **MODULES** | **STATUS** |
| 01 | INPUT(MAIN ROUTINE) | TO BE IMPLEMENTED |
| 02 | COUNT (SUBROUTINE) | TO BE IMPLEMENTED |
| 03 | PASSWORD(SUBROUTINE) | TO BE IMPLEMENTED |
| 04 | OUTPUT(MAIN ROUTINE) | TO BE IMPLEMENTED |

**02. INTRODUCTION**

Alphanumeric, also known as alphameric, simply refers to the type of Latin and Arabic characters representing the numbers 0 - 9, the letters A - Z (both uppercase and lowercase), and some common symbols such as @ # \* and &.

A best example is Sites requesting that you to create an alphanumeric password are asking us to use a combination of numbers and letters, which creates stronger passwords.

Alphanumeric passwords require both numbers and letters and special characters. The password should contain at least eight characters with one number, one special character and one lowercase and uppercase letter. If any of the above specifications are not satisfied, the password is not accepted and the Site notifies to define a new password with the required specifications.

**03. BASIC REQUIREMENT:**

|  |  |  |
| --- | --- | --- |
| **SNO** | **REQUIREMENT** | **STATUS** |
| 01 | A line of alphanumeric text | User defined |
| 02 | A specified pattern format | Standard |

**04. PROBLEM STATEMENT:**

Analyzing a Line of Alphanumeric Text: Analyze a line of alphanumeric text by examining each of the characters and display the total number of vowels, constants and digits in that line of text.

1) This can easily be accomplished by reading in a line of text, storing it in a one-dimensional

character array, and then analyzing the individual array elements.

2) The line of text should be read in main function, then analyzing the array elements in sub

function and result of counts should be print from main function

Example: “oiefr26a 3sdfi” while considering this line of alphanumeric text, vowels

5, constants 5, digits 3.

Also our machine can identify whether the given string is accepted as a standard password by any site.

1) This can be implemented by reading in a line of text, storing it in a one-dimensional

character array, and then analyzing the individual array elements.

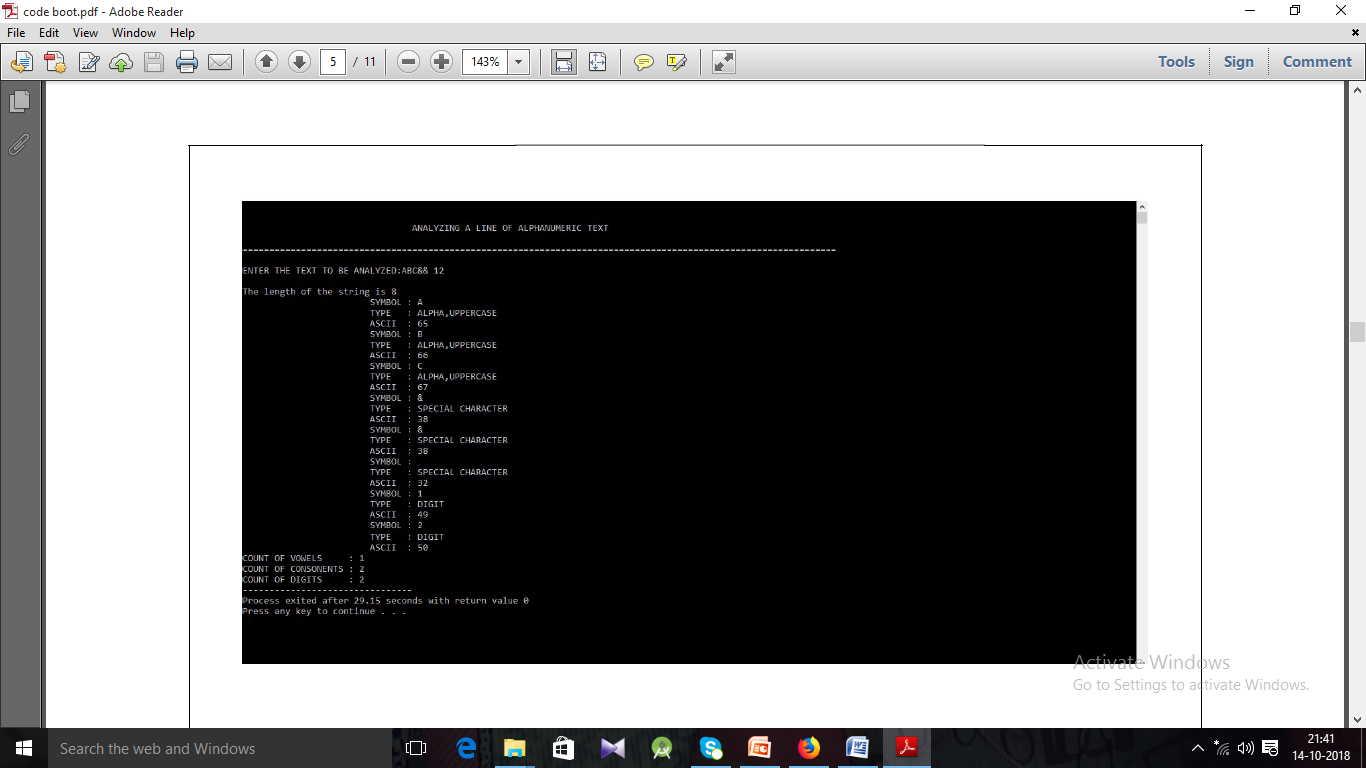
2) The line of text should be read in main function, then analyzing the array elements in sub

function and result should be printed.

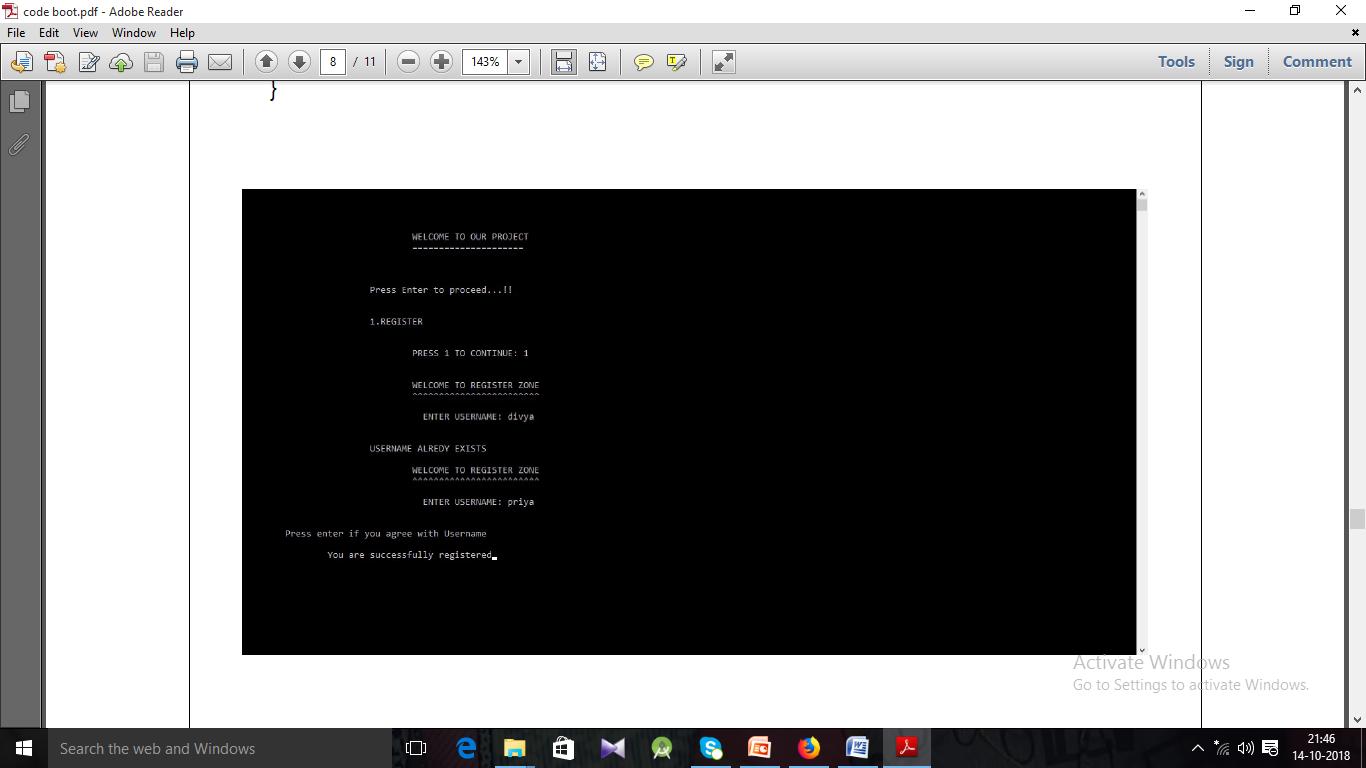
Example: “India\_2020” while considering this line of alphanumeric text given as a password, it is accepted. On the other hand “India” will not be accepted as it does not contain the minimum required specifications (i.e. India doesn’t neither have a special character nor a digit).

**8. SCREEN DISPLAY**

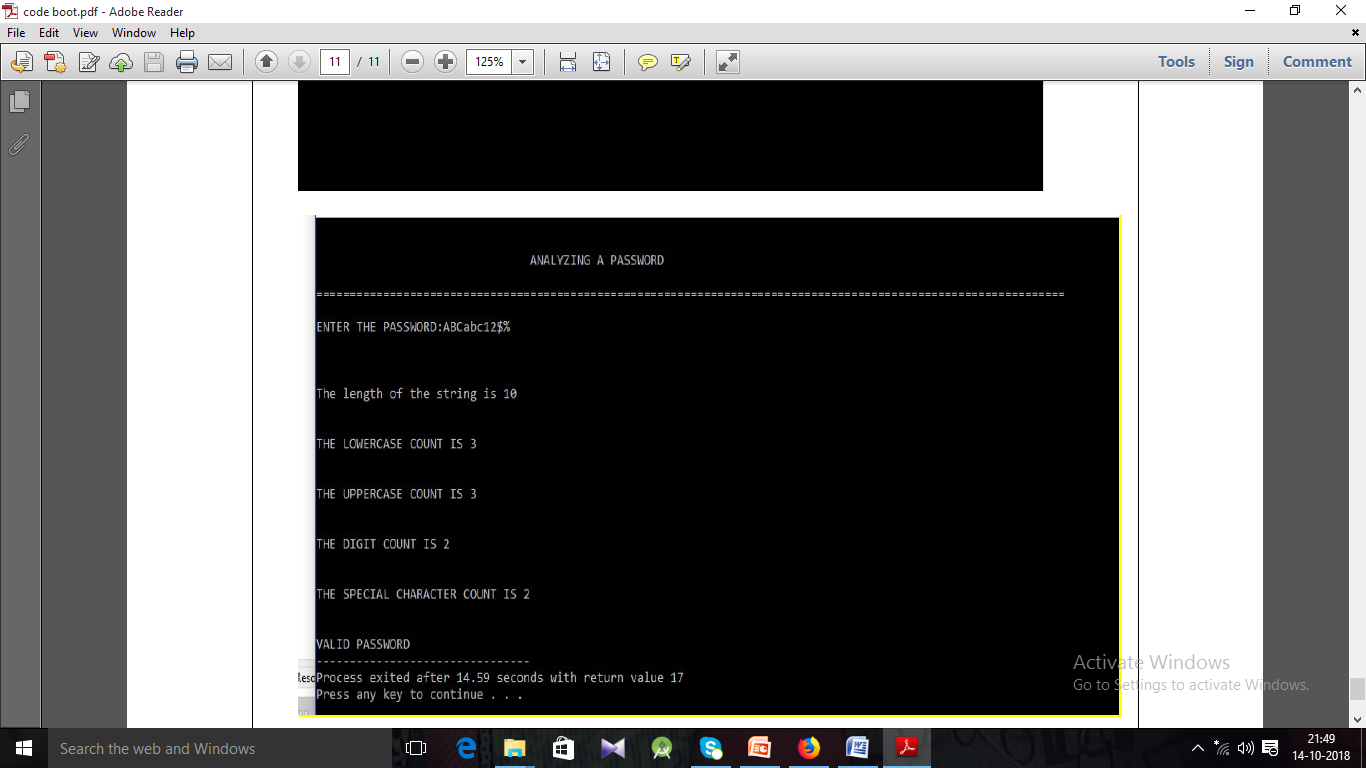
1. When an error message is displayed system should wait for a key press.

****

**FIG.8.1.MODULE 1,2**

****

**FIG.8.2MODULE 3**

**FIG.8.3.MODULE 4**

**9.LDD:**

**9.1 MAIN ROUNTINE**

**INPUTS:**

A line of alphanumeric text from user as hard declared input. There are no arguments involved in main function as we don’t get command line inputs and no external inputs are involved.

**SUBROUTINES CALLED:**

Count and password Check are the subroutines to be called in main function.

For count function, the base address of the character array containing the text is passed to the function.

**OUTPUT:**

1. The main analyses the input and prints the ASCII specifications of each character in the input.

2. In the above mentioned called subroutines, Count returns the count of vowels, consonants and digits which are printed in the main function. The main function returns nothing.

**9.2 SUBROUTINES:**

**1.COUNT:**

**INPUT:**

The base address of the array containing the text passed by the main function is received by a character pointer variable in the subroutine.

There are no subroutines called by this function.

**OUTPUT:**

The return type of subroutine is integer pointer. This subroutine is designed to count the number of vowels, consonants and digits. It stores them in an integer array and returns the base address of the integer array to the main function and the answers are printed in the main routine.

**2. PASSWORD CHECK:**

**INPUT:**

The base address of the array containing the text passed by the main function is received by a character pointer variable in the subroutine.

There are no subroutines called by this function.

**OUTPUT:**

This subroutine checks the specifications for a given string to be accepted as a strong password according to the standards. If the text satisfies the required standards, The string is accepted. If not an alert message to re-enter the password as per standard specifications is displayed.

**10. CONTROL FLOW DIAGRAM:**

Input alphanumeric text

Analyses input through ASCII

No

if input text matches the specifications

Specifications (array of [0-9a-zA-Z]+$&@/;)( sub routine call)

returns the base address of the array which stores the count of vowels, consonants and digits.

yes

and returns the base address of the array which stores the count of vowels, consonants and digits.

Main

Subroutine check password specification

no

yes

(print Password is accepted)

(Alert msg: password must have at least one special character, one number and one capital letter).

If satisfies specificationsnsnn

**11. PSUEDOCODE:**

**1.GET INPUT AND ASCII SPECIFICATIONS:**

1. Display “Enter a line of alphanumeric text”.
2. Input character array.
3. Process: Each element of the character array is analysed.
4. Output: ASCII descriptions are printed.

**2. COUNT:**

1. Input character array.
2. Analyze each element.
3. If element is between 0 to 9
   * + - 1. Increment digit count.
4. If element is ‘a’, ‘e’, ‘i’, ‘o’, ‘u’
   * + - 1. Increment vowel count.
5. If element is an alphabet other than vowel
   * + - 1. Increment consonant count.
6. Output: Display the count of digits, vowels, consonants.

**3. USER ID CREATION (VALIDITY CHECK):**

1. Input character array.
2. If the string already exists
   * + - 1. Then the name is not accepted and an alert message to enter new name is displayed.
3. Else accepts the string and displays an acknowledgement.

**4. PASSWORD GENERATION MODULE:**

1. Displays options to user
2. Only alphabets
3. Only digits
4. Alphabets and digits
5. Combination of alphabets, digits and special characters.
6. If user chooses option 1
   * + - 1. Then password containing only alphabets is generated and mailed to the user and acknowledgement is printed.
7. If user chooses option 2
   * + - 1. Then password containing only digits is generated and mailed to the user and acknowledgement is printed.
8. If user chooses option 3
   * + - 1. Then password containing digits and alphabets is generated and mailed to the user and acknowledgement is printed.
9. If user chooses option 4
   * + - 1. Then password containing combination of special characters, digits and alphabets is generated and mailed to the user and acknowledgement is printed.
10. Else the default statement which prints an alert message is displayed.

**5. PASSWORD VALIDITY MODULE:**

1. Input character array.
2. If the string does not satisfy password specifications
   * + - 1. Then the name is not accepted and an alert message to enter new name is displayed.
3. Else accepts the string and displays an acknowledgement.

**12. ALGORITHM:**

**1. GET INPUT AND ASCII SPECIFICATIONS:**

**STEP 1:** Have a character array to get the alphanumeric text as input.

**STEP 2:** Analyze the alphanumeric text.

**STEP 3:** Separate the symbols

**STEP 4:** If the symbol is a number display that it is a number and print the ASCII value of the symbol.

**STEP 5:**Similarly analyze each symbol and print the description.

**2. COUNT:**

**STEP 1:**The Alphanumeric line of text is fed as input to this module.

**STEP 2:**The symbols are segregated

**STEP 3:**The symbols are checked for digits, vowels and consonants.

**STEP 4:**Count the number of digits, vowels and consonants.

**STEP 5:**Display the number of digits, vowels and consonants.

**3.USER ID CREATION (VALIDITY CHECK):**

**STEP 1:**This module gets a string as input.

**STEP 2:**It checks if the string is already existing.

**STEP 3:**If it already exists, then the string entered is not accepted as username

**STEP 4:**Alert message to enter new username is displayed.

**STEP 1:**The new user name is again checked with the duplication.

**STEP 5:**And the process repeats until the user gives a valid username.

**4.PASSWORD GENERATION MODULE:**

**STEP 1:**This module asks the user about the requirements to generate a password.

**STEP 2:**For example if the user needs a password which contains only lowercase letters and digits, the machine generates a random password with the constraints.

**STEP 3:**Similarly the machine can generate passwords for the combinations only alphabets, alphabets and digits, only digits or even combination of alphabets, digits and special characters.

**STEP 4:**The password generated by this module is sent to the mail of the user and an acknowledgement is printed.

**5.PASSWORD VALIDITY MODULE:**

**STEP 1:**A line of alphanumeric text is given as input.

**STEP 2:**The input is verified with standard specifications of a strong password.

**STEP 3:**The standard specifications for a strong password insists the presence of at least eight symbols with one uppercase letter,one lowercase letter, one digit and one special character as mandatory.

**STEP 4:**If the password given by the user is not a strong password as per the specifications, an alert message to set a strong password is displayed.

**STEP 5:**If the password is strong as per the specifications, The password is accepted and an acknowledgement is printed.

.

**13.FLOW CHART:**

**GET INPUT ASCII SPECIFICATION:**

Start

Get input from user (i.e.) hard declared input

yes

no

If input is 0 to 9

Printf numbers, display its ASCII value

Else if input from ‘a’ to ‘z’

Else if input is from ‘A’ to ‘Z’

yes

Printf lowercase letter and display its ASCII value

no

Else if input is from ‘A’ to ‘Z’

Else if input is ‘&’ and ‘$’ or any special character

End

yes

no

Printf uppercase letter and display its ASCII value

Else if input is from ‘A’ to ‘Z’

yes

no

Printf special characters and display its ASCII

Error!!!!!

**USER ID VALIDITY CHECK**

START

Get a string as input from user (as hard declared input)

no

If name exists already

yes

The user name is accepted and acknowlegement is printed

Alert msg to enter new user

name

**PASSWORD GENERATION MODULE:**

Generate password containing only alphabets and mail it to client and print an acknowledgement

Case

Condition 1

Begin switch conditional expression

Choose option to generate password:

1) Only alphabets

2) Only digits

3) Digits and alphabets

4) Alpha, digits, special characters

Start

true

Case

Condition 2

true

Generate password containing only digits and mail it to client and print an acknowledgement

true

Case

Condition 2

Generate password containing digits and alphabets and mail it to client and print an acknowledgement

true

Generate password containing digits alphabets and special characters then and mail it to client and print an acknowledgement

Case

Condition 2

Default statement

**PASSWORD VALIDITY MODULE**

START

Get a string from the user as hard declared input

yes

Alert msg to enter new password

If password doesn’t satisfy specification

no

The password is accepted and acknowledgement is printed

**14.CODE:**

**GET INPUT AND ASCII SPCIFICATONS**

**COUNT**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int\* count(char\* ,int ); //function declaration

int main()

{

int i,length,cc=0,dc=0,vc=0;//cc->consonant, vc->vowel ,dc->digit count

int \*iptr; //pointer to receive address from function

char a[100]; //character array to get the text

printf("\n\n\t\t\t\tANALYZING A LINE OF ALPHANUMERIC TEXT\n\n================================================================================================================");

printf("\n\nENTER THE TEXT TO BE ANALYZED:");

scanf("\n%[^\n\t]s",a);

length=strlen(a);

printf("\nThe length of the string is %d",length);

for(i=0;i<length;i++)

{

if(a[i]>='a' && a[i]<='z') //to check if the symbol is lowercase letter

{

printf("\n\t\t\tSYMBOL : %c",a[i]);

printf("\n\t\t\tTYPE : ALPHABET,LOWERCASE");

printf("\n\t\t\tASCII : %d",a[i]);

/\* if(a[i]=='a' || a[i]=='e' || a[i]=='i' || a[i]=='o' || a[i]=='u')

vc++;

Else //commented lines

cc++;\*/

}

else if(a[i]>='A' && a[i]<='Z') //to check if the symbol is uppercase letter

{

printf("\n\t\t\tSYMBOL : %c",a[i]);

printf("\n\t\t\tTYPE : ALPHABET,UPPERCASE");

printf("\n\t\t\tASCII : %d",a[i]);

/\* if(a[i]=='A' || a[i]=='E' || a[i]=='I' || a[i]=='O' || a[i]=='U')

vc++;

Else //commented lines

cc++;\*/

}

else if(a[i]>='0' && a[i]<='9')

{

printf("\n\t\t\tSYMBOL : %c",a[i]);

printf("\n\t\t\tTYPE : DIGIT");

printf("\n\t\t\tASCII : %d",a[i]);

// dc++; commented line

}

else if(a[i]==' '||'&'||'\*'||'$') //checks if the symbol is special character

{

printf("\n\t\t\tSYMBOL : %c",a[i]);

printf("\n\t\t\tTYPE : SPECIAL CHARACTER");

printf("\n\t\t\tASCII : %d",a[i]);

}

}

iptr=count(&a,length); //function call by address

printf("\nCOUNT OF VOWELS : %d",\*(iptr));

printf("\nCOUNT OF CONSONENTS : %d",\*(iptr+1));

printf("\nCOUNT OF DIGITS : %d",\*(iptr+2));

return 0;

}

int\* count(char\* ch,int n) //function definition

{

int coun[3]={0,0,0}; //coun[0]-->vowel count

int i; //coun[1]-->consonant count

for(i=0;i<n;i++) //coun[2]-->digit count

{

if(ch[i]>='a' && ch[i]<='z')

{

if(ch[i]=='a' || ch[i]=='e' || ch[i]=='i' || ch[i]=='o' || ch[i]=='u')

coun[0]++; //increments vowel counter

else

coun[1]++; //increments consonant counter

}

else if(ch[i]>='A' && ch[i]<='Z')

{

if(ch[i]=='A' || ch[i]=='E' || ch[i]=='I' || ch[i]=='O' || ch[i]=='U')

coun[0]++; //increments vowel counter

Else

coun[1]++; //increments consonant counter

}

else if(ch[i]>='0' && ch[i]<='9')

{

coun[2]++; //increments digit counter

}

}

return (&coun); //returns the base address of the array containing counter variables

}

**USER ID VALIDATION CHECK**

#include<stdio.h>

#include<string.h>

static int i=0; //static global variable declaration

struct web //structure to store name and password of n users

{

char name[30],pass[30];

}w[99];

int n;

void reg(void); //function declaration

int main()

{

printf("\n\n\n\n\n\t\t\t\tWELCOME TO OUR PROJECT");

printf("\n\t\t\t\t=====================");

printf("\n\n\n\n\t\t\tPress Enter to proceed...!!");

if(getch()==13)

XY:

printf("\n\n\n\t\t\t1.REGISTER");

printf("\n\n\n\t\t\t\tPRESS 1 TO CONTINUE: ");

scanf("%d",&n);

switch(n)

{

case 1:

reg(); //function call

break;

default: printf("\n\n\t\t\t\tNO MATCH FOUND");

printf("\n\n\t\t\tPress Enter to re-Enter the choice");

if(getch()==13)

goto XY;

}

return 0;

}

void reg() //function definition

{

FILE \*fp; //file pointer declaration

char c,checker[30]; int z=0;

fp=fopen("user\_name.txt","ab+"); //creates file in append binary mode

printf("\n\n\t\t\t\tWELCOME TO REGISTER ZONE");

printf("\n\t\t\t\t^^^^^^^^^^^^^^^^^^^^^^^^");

for(i=0;i<100;i++)

{

printf("\n\n\t\t\t\t ENTER USERNAME: ");

scanf("%s",checker);

while(!feof(fp)) //termination condition

{

fread(&w[i],sizeof(w[i]),1,fp);

if(strcmp(checker,w[i].name)==0) //checks for duplicates

{

printf("\n\n\t\t\tUSERNAME ALREDY EXISTS");

reg();

}

else

{

strcpy(w[i].name,checker); //writes the name to structure

break;

}

}

// printf("\n\n\t\t\t\t DESIRED PASSWORD: ");

while((c=getch())!=13)

fwrite(&w[i],sizeof(w[i]),1,fp); //writes the name to file

fclose(fp);

printf("\n\n\tPress enter if you agree with Username");

if((c=getch())==13)

{

printf("\n\n\t\tYou are successfully registered");

}

break;

}

getch();

}

**PASSWORD VALIDATION CHECK**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

int i,length,lcc=0,dc=0,ucc=0,spcc; //lcc->lowercase count,ucc->uppercase

char a[100]; //dc->digit count,spcc->special character

printf("\n\n\t\t\t\tANALYZING A PASSWORD\n\n================================================================================================================");

printf("\n\nENTER THE PASSWORD:");

scanf("\n%[^\n\t]s",a);

length=strlen(a);

printf("\n\n\nThe length of the string is %d",length);

if(length>=8) //checks if the length of the password is //greater than or equal to 8 characters. {

for(i=0;i<length;i++)

{

if(a[i]>='a' && a[i]<='z')

{

lcc++; //increments lower case counter

}

else if(a[i]>='A' && a[i]<='Z')

{

ucc++; //increments uppercase counter

}

else if(a[i]>='0' && a[i]<='9')

{

dc++; //increments digit counter

}

else if(a[i]==' '||'&'||'\*'||'$')

{

spcc++; //increments special char counter

}

}

printf("\n\n\nTHE LOWERCASE COUNT IS %d",lcc);

printf("\n\n\nTHE UPPERCASE COUNT IS %d",ucc);

printf("\n\n\nTHE DIGIT COUNT IS %d",dc);

printf("\n\n\nTHE SPECIAL CHARACTER COUNT IS %d",spcc);

if(ucc>=1 && dc>=2 && spcc>=2 &&lcc>=1) //checks constraints

{

printf("\n\n\nVALID PASSWORD");

}

else

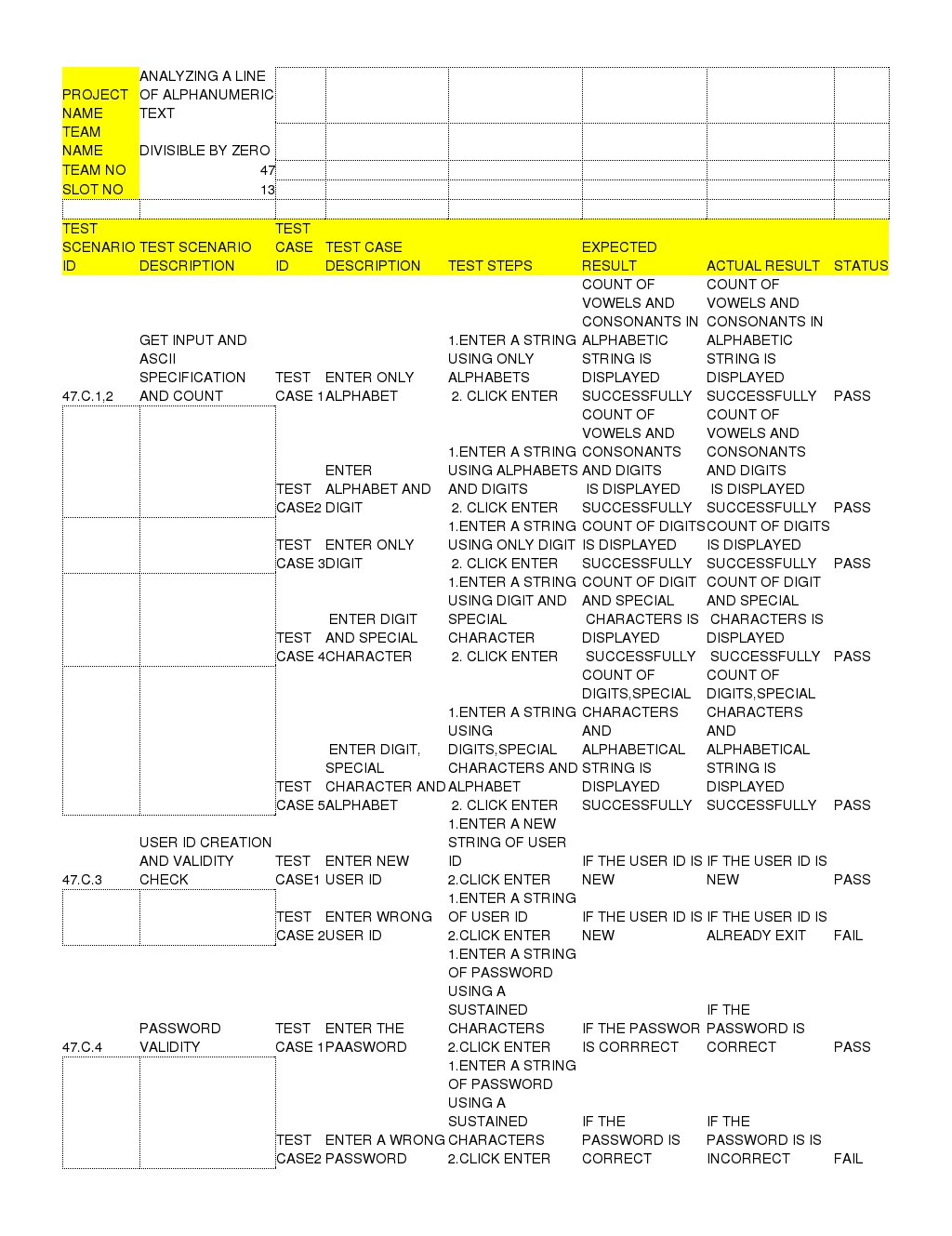
printf("\n\n\nINVALID PASSWORD");

}

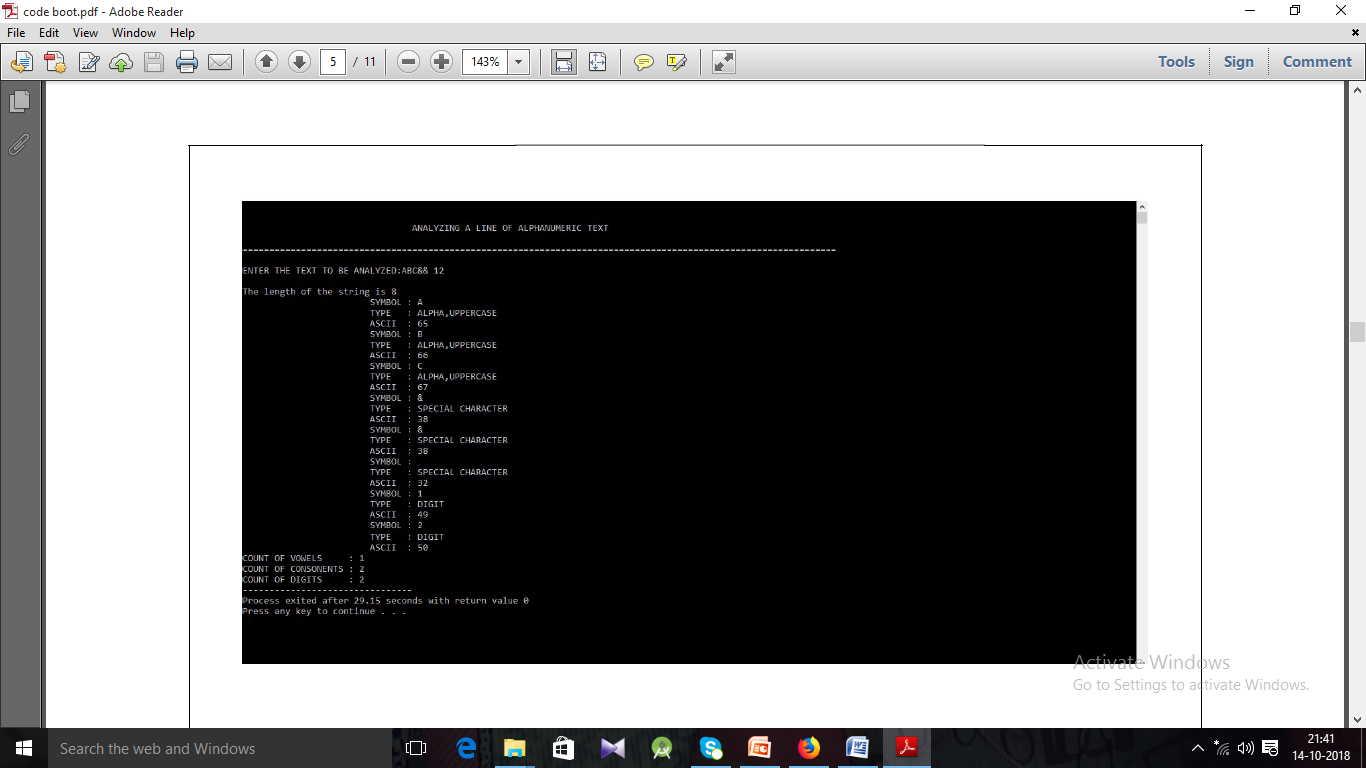
else

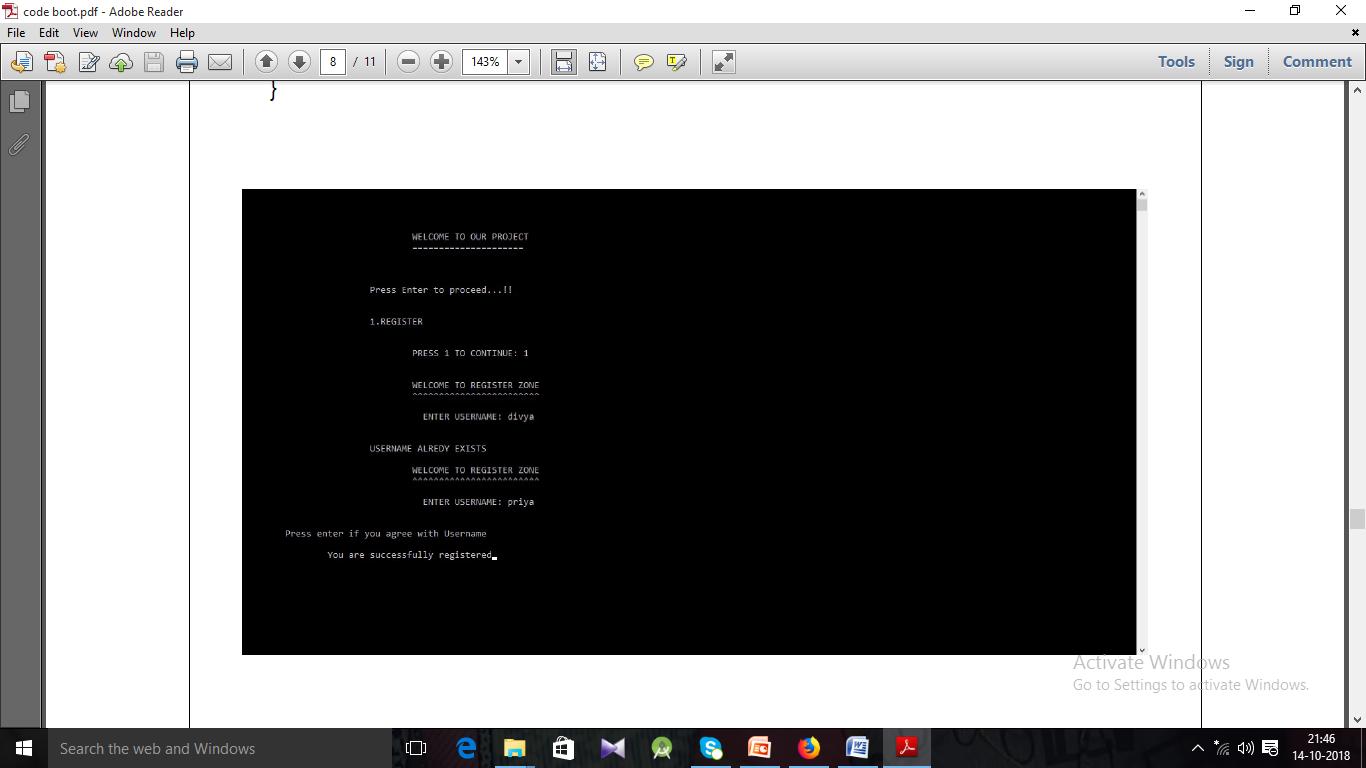
printf("\n\n\nINVALID PASSWORD");}

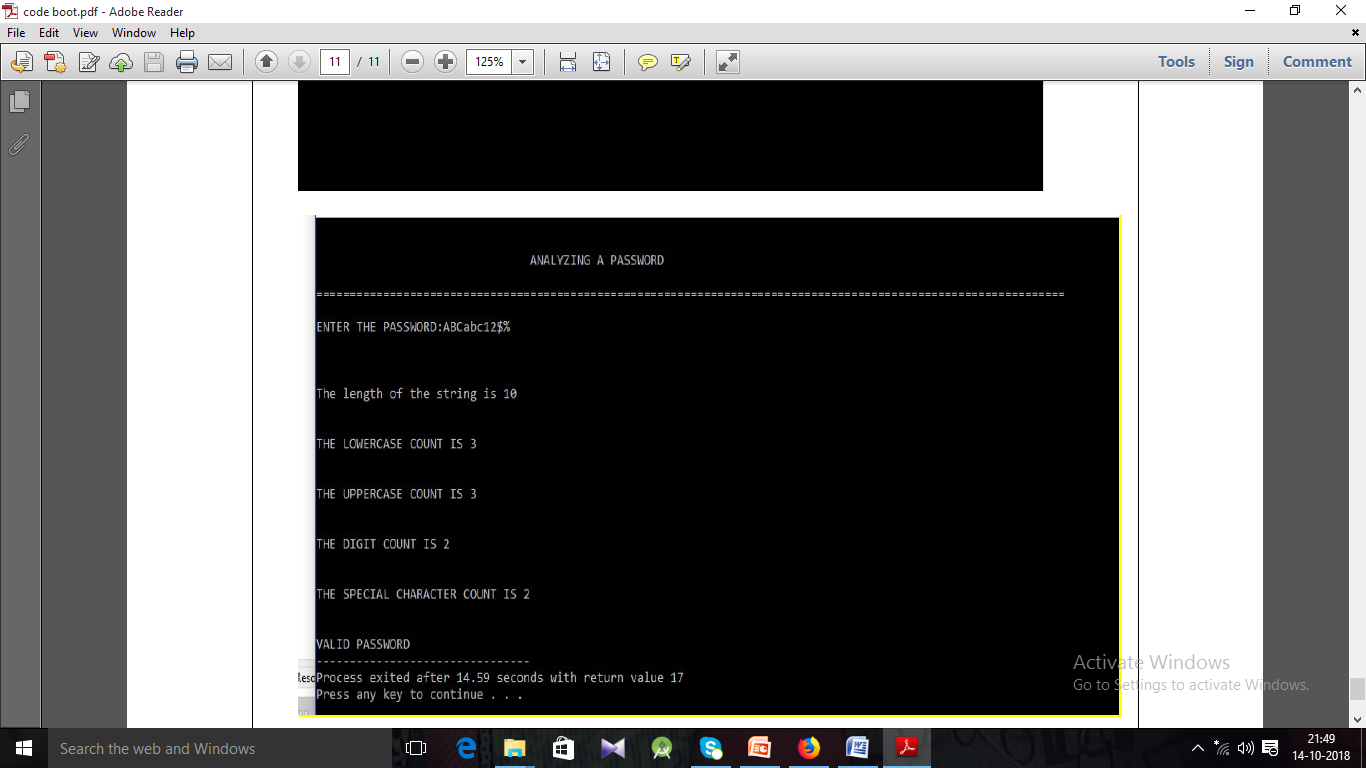
**15. TESTCASES**

****

**16. OUTPUT SCREENSHOT**



****

****

**17. COMPLEXITY:**

* For module 1 and 2 time complexity is 29.15 seconds
* For module 3 time complexity is 13 seconds
* For module 4 time complexity 14.59 seconds

**18. CONCLUSION:**

Thus our project analyze alphanumeric text and evaluates the count of requested features of text also includes eccentric applications of alpha numerals such as the validation of username by not accepting the duplicates and by validating an alphanumeric password according to standard norms.

**19.REFERENCES:**

*1.*https://study.com/academy/lesson/what-is-alphanumeric-definition-characters.html.

[2.https://stackoverflow.com/questions/10081492/analyse-format-of-alpha-numeric-string](https://stackoverflow.com/questions/10081492/analyse-format-of-alpha-numeric-string)

3.Arabic Handwritten Alphanumeric Character Recognition using Fuzzy Attributed Turning Functions.

DOI: 10.17877/DE290R-12648