**PES University, Bengaluru**

**Final Semester Assessment (FSA) – B.Tech. (CSE) – IV Sem**

**Session: JANUARY-MAY, 2019**

**UE17CS206 – Design and analysis algorithms**

Project Report

On

“Generate Password Suggestion”

Submitted by

|  |  |
| --- | --- |
| Nihal S | PES1201700188 |
| Abhilash H | PES1201701644 |

**Name of the Examiners Signature with Date**

1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Department of Computer Science & Engineering**

**PES University**

(Established under Karnataka Act No. 16 of 2013)

100 Feet Ring Road, Bengaluru – 560 085, Karnataka, India

**Abstract**

This project gives password suggestion to the given password type. More number of user accounts IS created as services increase. Each user account (of a particular person) having same or similar password may result in security breach. There are quite a few examples like these (having weak passwords) even in security fields. There should be a strong, non sequential password which also needs to be easy to remember. Thus combining user phrase with numbers and special characters is the best way.

**objective**

this project has many real world applications there is no account without password so generating a good password needs a priority to choose a strong password from wide range of passwords so to create a good password we need all the type of characters in it so that it can’t be hacked so easily by brute force hacking techniques this program takes an input of the password and generates similar kind of strong password.

**Code**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void insert\_substring(char \*a, char \*b, int position);

char \*substring(char \*string, int position, int length)

{

char \*pointer;

int c;

pointer = malloc(length+1);

if( pointer == NULL )

exit(EXIT\_FAILURE);

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

\*(pointer+c) = \*((string+position-1)+c);

\*(pointer+c) = '\0';

return pointer;

}

char\* add\_more\_char(char\* str, int need)

{

int pos = 0;

char c[1];

char low\_case[] = "abcdefghijklmnopqrstuvwxyz";

for (int i = 0; i < need; i++) {

pos = (rand() % strlen(str));

c[0] = low\_case[rand() % 26];

insert\_substring( str, c, pos);

}

return str;

}

void insert\_substring(char \*a, char \*b, int position)

{

char \*f, \*e;

int length;

length = strlen(a);

f = substring(a, 1, position - 1 );

e = substring(a, position, length-position+1);

strcpy(a, "");

strcat(a, f);

free(f);

strcat(a, b);

strcat(a, e);

free(e);

}

char\* suggester(int l,int u,int d,int s,char\* str){

char num[] = "0123456789";

char low\_case[] = "abcdefghijklmnopqrstuvwxyz";

char up\_case[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

char spl\_char[] = "@#$\_()!";

int pos =0;

char c[1];

if (l == 0) {

pos = (rand() % strlen(str));

c[0] = low\_case[rand() % 26];

insert\_substring( str, c, pos);

}

if (u == 0) {

pos = (rand() % strlen(str));

c[0] = up\_case[rand() % 26];

insert\_substring( str, c, pos);

}

if (d == 0) {

pos = (rand() % strlen(str));

c[0] = num[rand() % 10];

insert\_substring( str, c, pos);

}

if (s == 0) {

pos = (rand() % strlen(str));

c[0] = spl\_char[rand() % 7];

insert\_substring( str, c, pos);

}

return str;

}

void generate\_password(char \* pass){

int l=0;

int u=0;

int d=0;

int s=0;

int need=0;

int n = strlen(pass);

char suggest[200];

for(int i = 0; i < n; i++)

{

if (pass[i]>=97 && pass[i]<=122) {

l=1;

}

else if (pass[i]>=65 && pass[i]<=90){

u=1;

}

else if (pass[i]>=48 && pass[i]<=57){

d=1;

}

else

{

s=1;

}

// printf("%d %d %d %d\n",l,u,d,s);

}

if ((l + u + d + s) == 4) {

printf ("Your Password is Strong\n");

return;

}

else

{

printf ("Suggested passowrds \n");

}

for(int i = 0; i < 10; i++)

{

strcpy(suggest,suggester(l,u,d,s,pass));

// printf("%s",suggest);

need = 8 - strlen(suggest);

// printf("%d\n",need);

if (need >0)

{

printf("%d\n",need);

strcpy(suggest, add\_more\_char(suggest,need));

}

printf("%s\n",suggest);

}

}

int main()

{

char pass[200];

printf("Enter the password for which you need suggestions\n");

scanf ("%s",pass);

generate\_password(pass);

return 0;

}

Time Complexity of The Algorithm is O(n)

Basic Step of the Operation is comparing all the characters to set the character\_flags

***The whole algorithm is attached for reference:-***

Algorithm generate\_password (String Password)

{

//Input: Password (String) Which needs suggestions

//Output: Suggested passwords are printed

n <- length (Password)

lower,upper,digits,special\_symbols,need <- 0

//checking for lowercase ,uppercase, digits, special\_symbols

for i<-0 to n

{ if (Password contains small letters)

{

lower <- 1

}

if (Password contains upper letters)

{

upper <- 1

}

if (Password contains digits)

{

digits <- 1

}

if (Password contains special\_symbols)

{

special\_symbols <- 1

}

}

if (Password meets all the required constraints)

{

print "Password is strong"

return

}

else

{

print "The suggested passwords are:-"

}

// to get 10 passwords

for i <- 0 to 10

{

suggested\_password <- suggester(lower,upper,digits,special\_symbols)

//Assuming the default number of minimum characters is 8

need <- 8 - length(suggested\_password)

if (number of characters are less than 8 )

{

suggested\_password <- add\_more\_characters(suggested\_password,need)

}

print suggested Password

}

}

String suggester(int lower,int upper,int digits,int special\_symbols,Password)

{

//Input: flags and passwords

//Output: suggested\_password

numbers <- array of all digits

lowercase <- array of all lower alphabets

uppercase <- array of all upper alphabets

special\_characters <- array of all special symbols

int position <- 0

// Check for the flags and insert a random character of its type @ random position

if (Password doesnt have lower case )

{

position <- random integer of between 0 and 26

insert a random lowercase charatcer in that position in Password

}

if (Password doesnt have upper case )

{

position <- random integer of between 0 and 26

insert a random uppercase charatcer in that position in Password

}

if (Password doesnt have a digit)

{

position <- random integer of between 0 and 10

insert a random number charatcer in that position in Password

}

if (Password doesnt have special\_character)

{

position <- random integer of between 0 and 7

insert a random special\_character charatcer in that position in Password

}

return Password

}

// If the suggester fails to give 8 number of characters than we have to add some characters to it

String add\_more\_characters(String Password, int need)

{

position <- 0

// put anyone of lowercase characters in the Password

lowercase <- array of all lower case characters

for i <- 0 to need

{

insert random lowercase alphabets to random position on Password

}

return Password

}