GCSE Computer Science Component 3 NEA
Veer Vohra

Introduction

This program does both these tasks:
> Check if the user's password is good, determine its strength and give it a score
> Generate a strong password

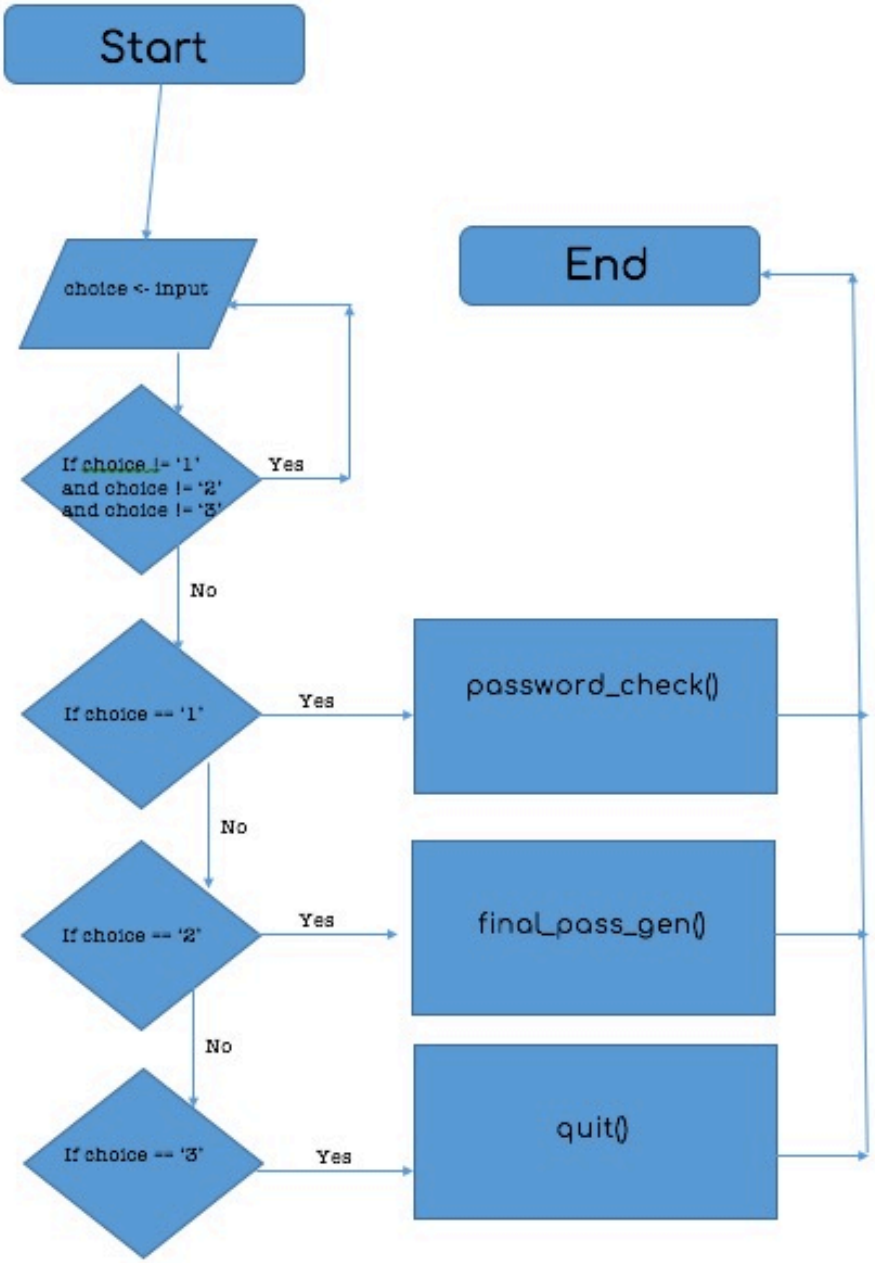
Aims/Objectives for this program:

> The menu should be appealing and easy to use
> The password checking should be accurate and efficient
> The password generator should produce strong passwords
> The entire program should be efficient
> The entire program should be user friendly and easy to use

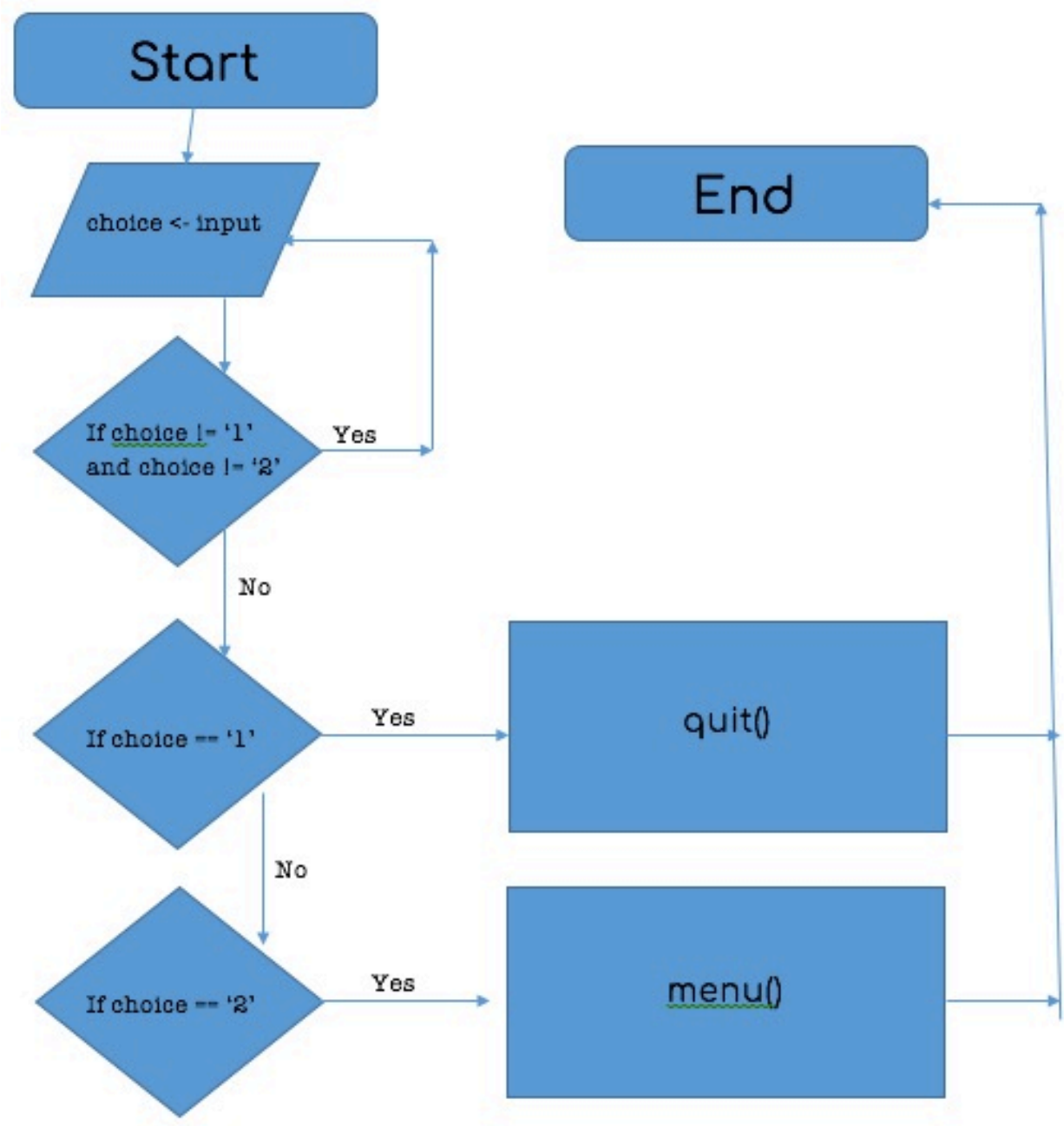
Design

- The entire program will be split into subroutines to ensure that it is efficient and organized.
- The data types used are:
    > Strings
    > Integers
    > Arrays/Lists
- Data validation has been taken care of within each function
    > This involves ensuring that the user has entered normal data. Erroneous data will be rejected and the user will be prompted to enter another input

```
---------- Main Menu ----------
FUNCTION menu()
    OUTPUT ("Welcome To 2Password\n1 - Check Password\n2 - Generate
    Password\n3 - Quit\n>>> ")
    choice <- INPUT
    while choice != '1' AND choice != '2' AND choice != '3':
        choice <- INPUT("ERROR\n>>> ")
    ENDWHILE
    IF choice = '1' THEN
        password_check()
    ELSEIF choice = '2' THEN
        final_pass_gen()
    ELSEIF choice = '3' THEN
        OUTPUT "Thank you for using 2Password"
        quit()
    ENDIF
ENDFUNCTION
```

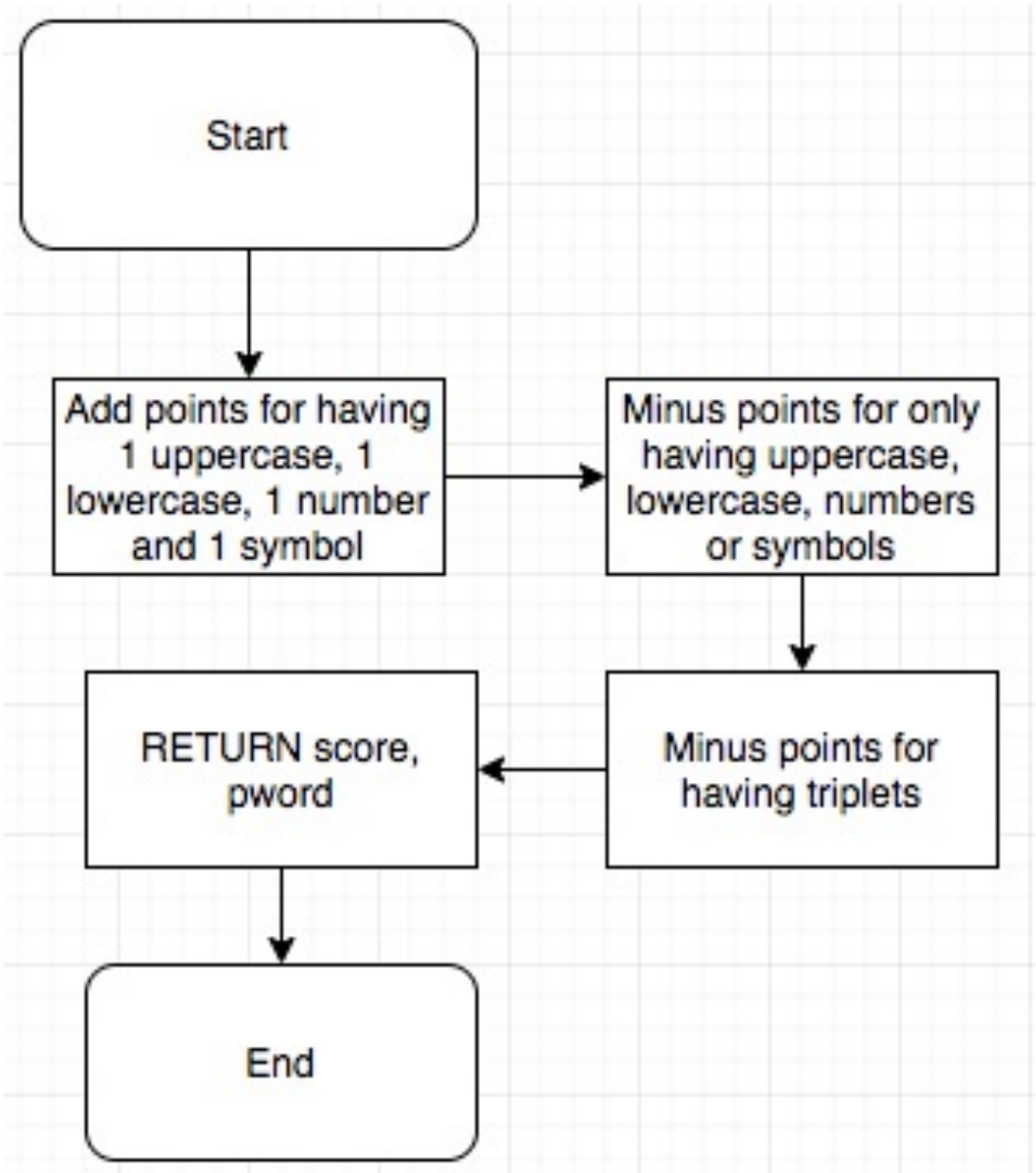


```
---------- Returning to the Menu or Exiting ----------
FUNCTION end_of_task()
    now_what <- INPUT("\n1 - Exit\n2 - Main Menu\n>>> ")
    while now_what != '1' AND now_what != '2'
        now_what <- INPUT("ERROR\n>>> ")
    ENDWHILE
    IF now_what = '1' THEN
        OUTPUT "Thank you for using 2Password"
        quit()
    ELSE
        menu()
    ENDIF
ENDFUNCTION
```

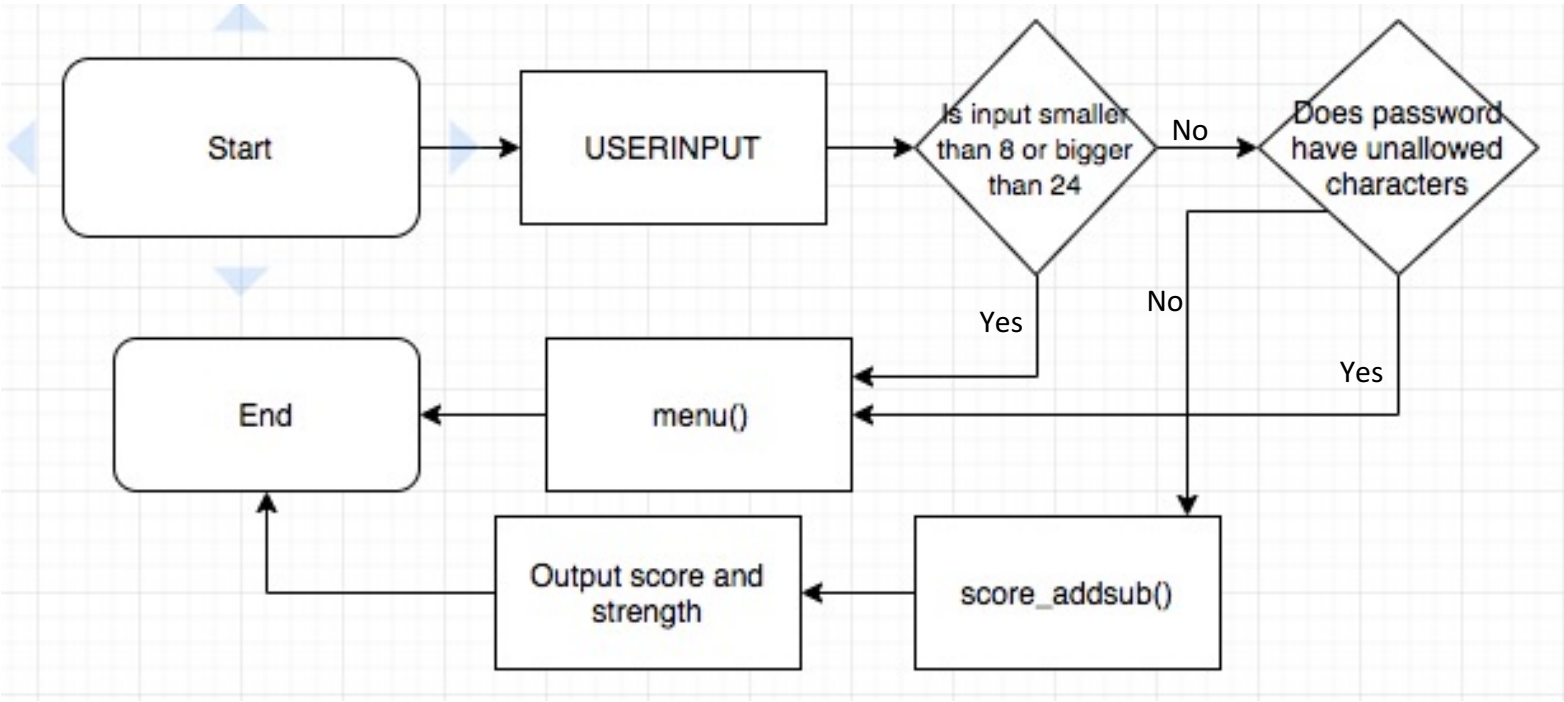


```
---------- Calculating the score ----------
FUNCTION score_addsub(pword,p_list)
    triplet_in_p <- []
    digits, symbols, score, letters, bonus <- 0, 0, len(pword), 0, 0
    upcase, lowcase, number, symbol <- False, False, False, False
    triplets <- ["qwe", "wer", "ert", "rty", "tyu", "yui", "uio",
"iop","asd", "sdf", "dfg", "fgh", "ghj", "hjk", "jkl","zxc", "xcv", "cvb",
"vbn"]
    FOR char IN p_list:
        IF 65 <= ord(char) <= 90 AND not upcase THEN
            score <- score + 5
            bonus <- bonus + 1
            upcase <- True
        ELSEIF 97 <= ord(char) <= 122 AND not lowcase THEN
            score <- score + 5
            bonus <- bonus + 1
            lowcase <- True
        ELSEIF char.isdigit() AND not number THEN
            score <- score + 5
            bonus <- bonus + 1
            number <- True
        ELSEIF char in p_list AND char.isdigit() = False AND
char.isalpha() = False AND not symbol THEN
            score <- score + 5
            symbol <- True
        ENDIF
    ENDFOR
    IF bonus = 4 THEN
        score += 10
    ENDIF
    FOR char IN p_list
        IF char.isdigit()THEN
            digits <- digits + 1
        ELSEIF char.isalpha()THEN
            letters <- leters + 1
        ELSEIF char in p_list AND char.isdigit() = False AND
char.isalpha() = False THEN
            score <- score + 5
        ENDIF
    ENDFOR
    IF digits <- len(pword) THEN
        score <- score - 5
    ENDIF
    IF letters = len(pword) THEN
        score <- score - 5
    ENDIF
    IF symbols = len(pword) THEN
```

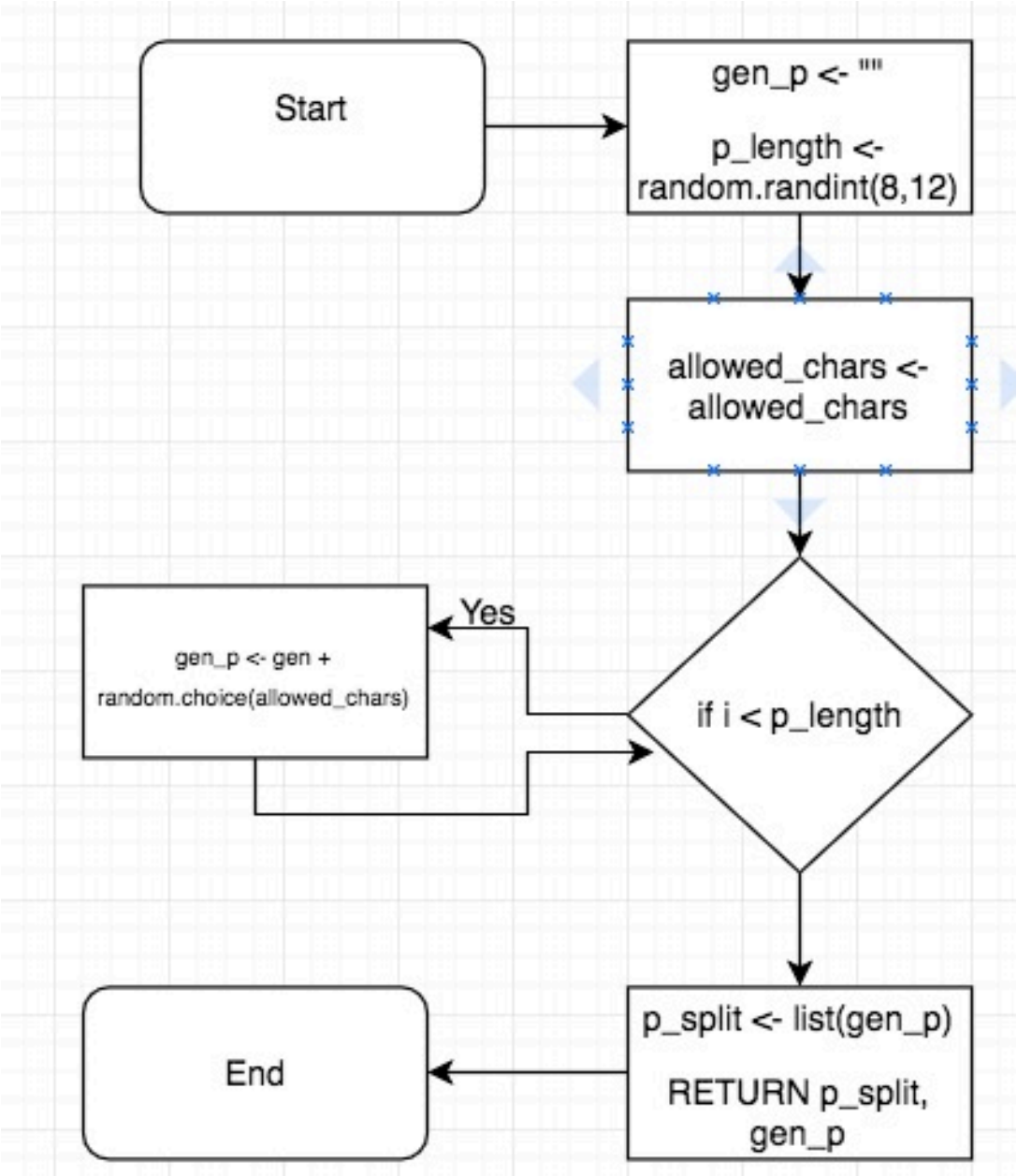
```
        score <- score — 5
    ENDIF

    FOR triplet IN triplets
        IF triplet IN pword.lower()THEN
            triplet_in_p.append(triplet)
        ENDIF
    ENDFOR
    score <= score - (length(triplet_in_p) * 5)
    RETURN score, pword
ENDFUNCTION
```



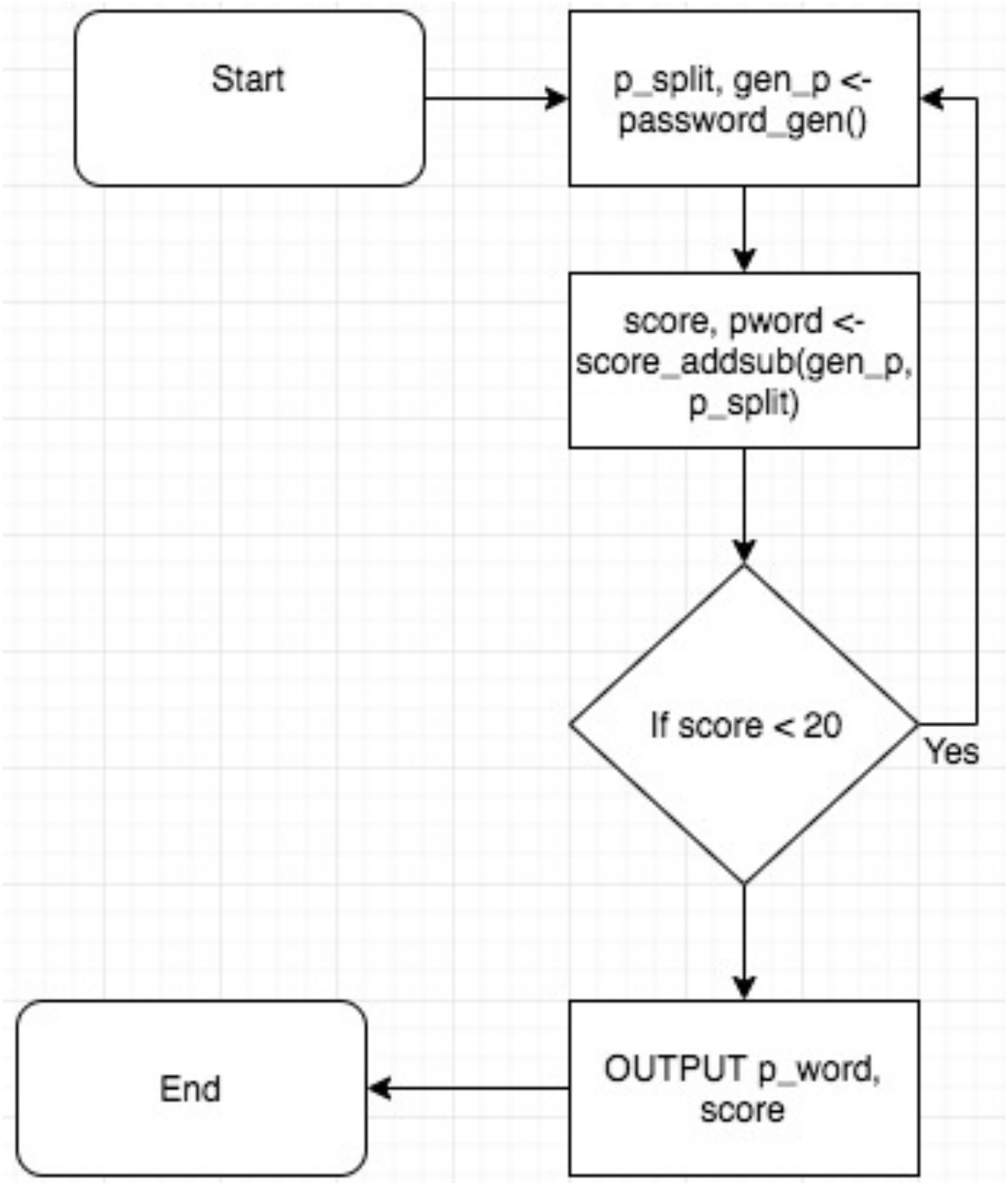
```
---------- Checking if the entered password is allowed ----------
FUNCTION password_check()
    allowed_chars <-
["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","
S","T","U","V","W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k"
,"l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","0","1","2","
3","4","5","6","7","8","9","!","$","%","^","&","*","(",")","_","-","+","<-
"]
    rejected <- []
    pword <- input("Please enter your password\n>>> ")
    WHILE length(pword) < 8
        pword <- INPUT("Password must be longer that 8 characters\n>>> ")
    ENDWHILE
    WHILE len(pword) > 24
        pword <- INPUT("Password must be shorter that 24 characters\n>>>
")
    ENDWHILE
    p_list <- list(pword)
    FOR char IN p_list
        IF char NOT IN allowed_chars:
            rejected.append(char)
        ENDIF
    ENDFOR

    WHILE length(rejected) > 0
        OUTPUT "Your password contains unallowed chars\nThese are :
",rejected
        pword <- input(">>> ")
        p_list <- pword.split()
        rejected <- []
        FOR char IN p_list
            IF char not IN allowed_chars THEN
                rejected.append(char)
            ENDIF
    ENDWHILE
    score, pword <- score_addsub(pword,p_list)
    OUTPUT "Your password : ", pword
    IF score >= 20 THEN
        OUTPUT "Your password score is : ", score, "\nSTRONG"
    ELSEIF score <= 0 THEN
        OUTPUT "Your password score is : ", score, "\nWEAK"
    ELSE
        OUTPUT "Your password score is : ", score, "\nMEDIUM"
    ENDIF
    end_of_task()
ENDFUNCTION
```


Start
USERINPUT
Is input smaller than 8 or bigger than 24
No
Does password have unallowed characters
Yes
No
Yes
End
menu()
Output score and strength
score_addsub()

```
---------- Generating a random password ----------
FUNCTION password_gen()
    gen_p <- ""
    p_length <- random.randint(8,12)
    allowed_chars <-
["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","
S","T","U","V","W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k"
,"l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","0","1","2","
3","4","5","6","7","8","9","!","$","%","^","&","*","(",")","_","-","+","<-
"]
    FOR i IN range(p_length)
        gen_p <- gen + random.choice(allowed_chars)
    ENDFOR
    p_split <- list(gen_p)
    RETURN p_split, gen_p
ENDFUNCTION
```



```
---------- Printing the generated password and its score ----------
FUNCTION final_pass_gen():
    p_split, gen_p <- password_gen()
    score, pword <- score_addsub(gen_p, p_split)
    WHILE score < 20
        p_split, gen_p <- password_gen()
        score, pword <- score_addsub(gen_p, p_split)
    ENDWHILE
    OUTPUT "Your password : ", pword,"\nYour password score is : ", score,
"\nSTRONG"
    end_of_task()
ENDFUNCTION
```



Code

- The code for this program has been saved in a separate file.
- It has been written in Python 3.6.0

```
-------------------- CODE --------------------
import passGen as pGen
import password_check as pCheck

def menu():
    choice = input("----------------------------------------\nWelcome To
2Password\n1 - Check Password\n2 - Generate Password\n3 - Quit\n>>> ")
    while choice != '1' and choice != '2' and choice != '3': # data
validation
        choice = input("ERROR\n>>> ")
    print("--------------------")
    # checking user choice and directing them to desired function
    if choice == '1':
        pCheck.password_check()
        end_of_task()
    elif choice == '2':
        pGen.final_pass_gen()
        end_of_task()
    elif choice == '3':
        print("\n\n--------------------\nThank you for using 2Password\n--
------------------")
        quit() # quitting

def end_of_task():
    # ---------- Menu/Exit ---------- #
    now_what = input("\n----------------------------------------\n1 -
Exit\n2 - Main Menu\n>>> ")
    while now_what != '1' and now_what != '2':
        now_what = input("ERROR\n>>> ")
    print("--------------------")
    if now_what == '1':
        print("\n\n--------------------\nThank you for using 2Password\n--
------------------")
        quit()
    else:
        menu()

menu()



def score_addsub(pword,p_list):
```

```
    # ---------- Variables ---------- #
    triplet_in_p = []
    digits, symbols, score, letters, bonus, upcase, lowcase = 0, 0,
len(pword), 0, 0, 0, 0
    upcase, lowcase, number, symbol = False, False, False, False
    triplets = ["qwe", "wer", "ert", "rty", "tyu", "yui", "uio",
"iop","asd", "sdf", "dfg", "fgh", "ghj", "hjk", "jkl","zxc", "xcv", "cvb",
"vbn","bnm"]

    # ---------- Adding points for having
uppercase/lowercase/numbers/symbols ---------- #
    for char in p_list:
        if 65 <= ord(char) <= 90 and not upcase: # if it is an uppercase
letter
            score += 5
            bonus += 1
            upcase = True
        elif 97 <= ord(char) <= 122 and not lowcase: # if it is a
lowercase letter
            score += 5
            bonus += 1
            lowcase = True
        elif char.isdigit() and not number: # if its a number
            score += 5
            bonus += 1
            number = True
        elif char in p_list and char.isdigit() == False and char.isalpha()
== False and not symbol: # if it is a symbol
            score += 5
            symbol = True
    if bonus == 4: # checking if they have achieved the bonus
        score += 10

    # ---------- Subtracting points for having only
symbols/letters/numbers ---------- #
    for char in p_list: # iterating through password
        if char.isdigit():
            digits += 1
        elif char.isalpha():
            letters += 1
            if char.upper() == char:
                upcase += 1
            elif char.lower() == char:
                lowcase += 1
        elif char in p_list and char.isdigit() == False and char.isalpha()
== False:
            symbols += 1
```

```
    if letters == len(pword) or symbols == len(pword) or digits ==
len(pword): # if there are only numbers/letters/symbols
        score -= 5
    if symbols == 0 and digits == 0 and upcase > 0 and lowcase == 0 or
symbols == 0 and digits == 0 and upcase > 0 and lowcase > 0 or symbols ==
0 and digits == 0 and upcase == 0 and lowcase > 0:
        # if there are only uppercase or only lowercase
        score -= 5

    # ---------- Subtracting points for keyboard patterns ---------- #
    for triplet in triplets:
        if triplet in pword.lower():
            triplet_in_p.append(triplet)
    score -= (len(triplet_in_p) * 5)

    return score, pword


import score as pScore
import random

def password_gen():
    gen_p = ""
    p_length = random.randint(8,12)
    allowed_chars =
["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","
S","T","U","V","W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k"
,"l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","0","1","2","
3","4","5","6","7","8","9","!","$","%","^","&","*","(",")","_","-
","+","="]
    for i in range(p_length):
        gen_p += random.choice(allowed_chars)
        p_split = list(gen_p)
    return p_split, gen_p

def final_pass_gen():
    p_split, gen_p = password_gen()
    score, pword = pScore.score_addsub(gen_p, p_split)
    while score < 20:
        p_split, gen_p = password_gen()
        score, pword = pScore.score_addsub(gen_p, p_split)
    print("Your password : ", pword,"\nYour password score is : ", score,
"\nSTRONG")


import score as pScore
```

```
def password_check():
    # ---------- Variables ---------- #
    allowed_chars =
["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","
S","T","U","V","W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k"
,"l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","0","1","2","
3","4","5","6","7","8","9","!","$","%","^","&","*","(",")","_","-
","+","="]
    rejected = []
    pword = input("Please enter your password\n>>> ")

    # ---------- Checking length ---------- #
    if len(pword) < 8:
        print("Password must be longer that 8 characters")
        import main as m
        m.menu()
    elif len(pword) > 24:
        print("Password must be shorter that 24 characters")
        import main as m
        m.menu()

    # ---------- Checking for unallowed characters ---------- #
    p_list = list(pword)
    for char in p_list:
        if char not in allowed_chars:
            rejected.append(char)
    if len(rejected) > 0:
        print("Your password contains unallowed chars\nThese are :
",rejected)
        menu()

    score, pword = pScore.score_addsub(pword,p_list)

    # ---------- Printing score ---------- #
    print("Your password : ", pword)
    if score >= 20:
        print("Your password score is : ", score, "\nSTRONG")
    elif score <= 0:
        print("Your password score is : ", score, "\nWEAK")
    else:
        print("Your password score is : ", score, "\nMEDIUM")
```

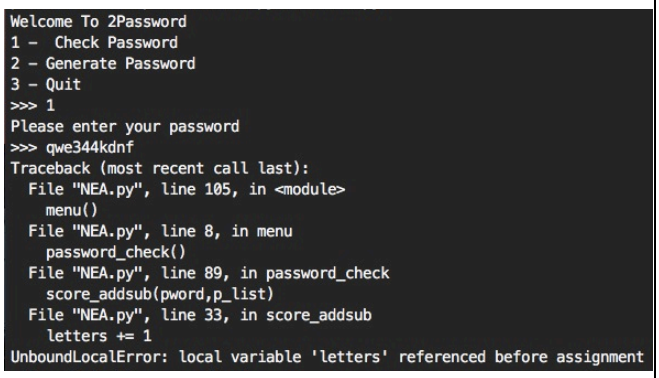
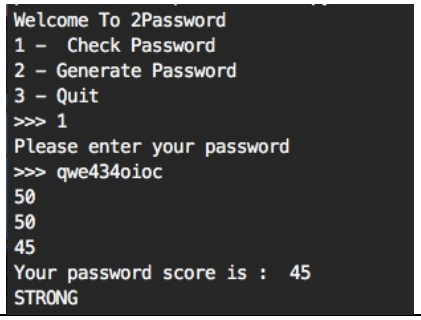
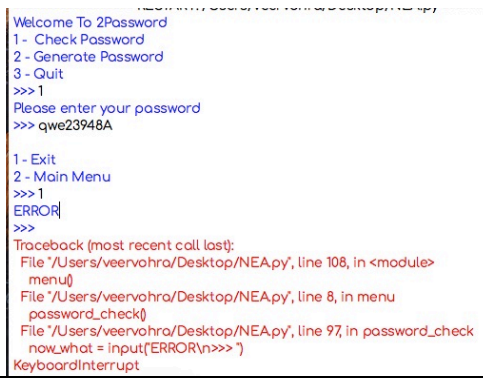
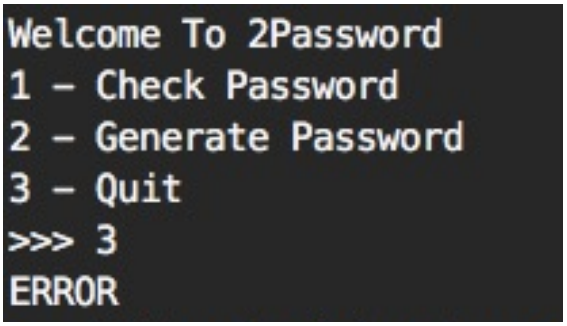
Testing

- A table of tests has also been attached to the end of the file.
- It includes normal, extreme and erroneous data to confirm the robustness of the program.

Variables

Variable Name	Data Type	Validation
choice	String	Used WHILE loop to make sure choice is 1,2 or 3.
pword	string	Used the function password_check to make sure entered pword met the requirements.
plist	string	Is just a listed version of *pword* so its already valid.
now_what	string	Used WHILE loop to make sure choice is 1 or 2.
triplet_in_p	List of strings	N/A
digits	integer	N/A
symbols	integer	N/A
score	integer	N/A
letters	integer	N/A
bonus	integer	N/A
allowed_chars	List of characters	N/A
gen_p	string	N/A
p_split	List of characters	N/A

---------- Table with Trials and Errors ----------

Test Number	Test Description	Test Data	Test Output	Comments
1	Testing while coding.	1		The variable "letters" had been referenced before being defined.
2	Testing while coding	1 qwe434oioc		The score for the triplet had not been deducted
3	Testing while coding	1 qwe23948A 1		After checking the password, it didn't registed the option to exit the program
4	Testing while coding. This shouldn't output "ERROR" because '3' is an option.	3		The program didn't recognize the choice "3"

5	Testing the main menu inputs	5
		4
		6
		3
			---------- 	
			Welcome To 2Password
			1 - Check Password
			2 - Generate Password
			3 - Quit
			>>> 5
			ERROR
			>>> 4
			ERROR
			>>> 6
			ERROR
			>>> 10000
			ERROR
			>>> 3
			---------- 	
			
			---------- 	
			Thank you for using 2Password
			---------- 	
			>>>
				It works flawlessly

6	Testing the password generation	2	---------- Welcome To 2Password 1 - Check Password 2 - Generate Password 3 - Quit >>> 2 ---------- Your password : q()%Z-S9 Your password score is : 28 STRONG ---------- 1 - Exit 2 - Main Menu >>>	It works flawlessly
7	Testing the validation of password entry	1	Please enter your password >>> 2 Password must be longer that 8 characters Traceback (most recent call last): File "/Users/veervohra/Desktop/NEA/code/main.py", line 32, in <module> menu() File "/Users/veervohra/Desktop/NEA/code/main.py", line 11, in menu pCheck.password_check() File "/Users/veervohra/Desktop/NEA/code/password_check.py", line 12, in check menu() NameError: name 'menu' is not defined >>>	The function "menu" is in a separate file which wasn't defined

Evaluation

- My code has met my objectives because it is easy to use, user friendly and robust
- To improve my code, I would incorporate a Graphical User Interface (GUI) using TKinter, Pygame or incorporate it into a website using HTML, CSS and Javascript
- The problems I encountered in the program were minimal. The main problem I experienced was a minor error in the menu which caused every entry to output "ERROR". I solved this problem by reading through the code and re-structuring it.