Project 2 (C++): You are to implement an inter-active new password checker.

First, your program displays the password rules (see below) and then asks the user to create a new password.

The list of password rules are:

1) The password length: 8-32 characters
2) At least one numerical, i.e., 0, 1, 2,..
3) At least one upper case letter, i.e., A, B, C, ...
4) At least one lower case letter, i.e., a, b, c, ...
5) At least one of the special characters, but it must be within the set:{ # $ * ( ) % & ^} a total of eight (8) special characters. no other special characters are allowed.

After the user typed his/her password, your program checks the validity of new password against the password rules.
If the new password is NOT valid, your program will display the password rules and ask the user to create a new password all over again as at the beginning.
If the user's new password is valid, then, your program will ask the user to re-type the new password for the second time to see if it matches the original typed password; if it matches, then your program will say "your password will be updated in a few minutes" and exit; if it does NOT match, your program will inform the user to re-enter again, after three attempt, your program will ask the user to create a new password all over again as at the beginning.

*** Run your program and enter the following fail cases:

1) password is shorter than 8
2) password is long than 32
3) password contains one or two special symbols that are not legal
4) password does not contain at least one numerical
5) password does not contain at least one upper case letter
6) password does not contain at least one lower case letter
7) password does not contain at least one of legal special characters

*** Your hard copy *.pdf includes:
- cover sheet
- source code
- print outFile

****
Language: (C++) // C++ tutorial & CPP dev tutorial are posted on Google Classroom
****
Project points: 10 pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):
9/7/2020 Monday before midnight
-1 for 1 day late: 9/8/2020 Monday before midnight
-2 for 2 days late: 9/9/2020 Tuesday before midnight
-10/10 : after 9/9/2020 Tuesday after midnight

*** Name your soft copy and hard copy files using the naming convention as state in “Project Submission Requirement” posted on Google Classroom.

*** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in the same email attachments with correct file names as state in “Project Submission Requirement”; otherwise, it would not count as submission.

```
*******************************
I. Input: inter-active from console by user

*******************************
II. Output:
      a) inter-active displays on screen
      b) outFile (use argv[1]): This file is used to record the screen display.

*******************************
III. Data structure:
*******************************

 - A passWordChecker class

      - password (string) // enter by the user

      - secondPassword (string) // enter by the user

      - passwordLength (int) // strlen of the password

      - charCount[5] (int) // an 1D integer array to store the counts of each of
                 the password rules:
                       charCount[0] store the count of illegal special characters
                       charCount[1] store the count of numeric
                       charCount[2] store the count of lower case characters
                       charCount[3] store the count of upper case characters
                       charCount[4] store the count of legal special characters

                       // intialize to zero

       methods: All methods are on your own,

      - constructor () // initialize all members accordingly

      - displayRules (outFile) // displays on the console screen and output to
                              outFile: "Create your password" and the password rules

      - askPasswd (outFile) // displays on the console screen and output to outFile
                              "Please enter your password"

      - displaySucess (outFile) // displays on the console screen and output to
                              outFile: "your password will be updated in a few
                              minutes"

      - displayFail(outFile) // displays on the console screen and output to
                              outFile: "your password failed one or more password
                              rules"

       - displayMatchFail(outFile) // displays on the console screen and output to
                              outFile: "Match fail... "
```

- checkCharType (ch) // Check ch to see what it is; i.e., numeric, upper case, lower case, which special character, it returns an index 0, 1, 2, 3, or 4, accordingly.
  Use the asci code of ch for easy indexing!!!!
  This is the only method in this project that is a bit challenging to code! Be more creative on the data structures use in this method. This is a good interview question!

- checkRules() // this method checks each entry of charCount array to determine the validity of the password, ie,
  // charCount [0] is 0
  // charCount [1] to charCount [4] are greater than 0
  // if all valid, it returns 1, otherwise, returns 0.

- matching (s1, s2) // this method checks to see if s1 and s2 are matched exactly; if two match, it returns 1, otherwise, it returns 0

```
*******************************
IV. main ()
*******************************
Step 0: outFile ← open argv[1] to write

Step 1: displayRules (outFile)

Step 2: askPasswd (outFile)
        password ← screen input from the user
        passwordLength <-- the length of password (use strlen)

step 3: repeat step 2 if the length of password is NOT within the range of 8 - 32
        print error message

step 4: i ← 0

step 5: index ← checkCharType(password[i]) // make sure the index is within 0 - 4
        charCount[index] ++

step 6: i++

step 7: repeat step 5 to step 6 until the last password character is checked.

step 8: validYesNo ← checkRules ()
        if validYesNo is not good (!= 1)
           call displayFail(outFile)

step 9: repeat step 1 to step 8 if validYesNo is 0

step 10: display and ask user to re-type his/her password; also write to outFile

step 11: secondPassword ← from the user

step 12: matchYesNo ← matching (password, secondPassword)

step 13: if matchYesNo is no good (== 0)
            displayMatchFail(outFile)

step 14: repeat step 1 to step 13 until matchYesNo == 1

step 15: call displaySucess (outFile)
```