**Programming Assignment: Random Password Generator**

**Objective**

Create a program that generates a secure, random password based on user preferences. The user can customize the length of the password and the types of characters to include.

**Learning Goals**

By completing this assignment, you will:

1. Work with randomization to generate unpredictable sequences.
2. Use conditionals to handle user preferences.
3. Practice string manipulation and input validation.

**Assignment Requirements**

**1. Input**

* Prompt the user to specify the password length.
  + Validate the input to ensure it is a positive integer.
* Ask the user what types of characters to include:
  + Uppercase letters (A-Z).
  + Lowercase letters (a-z).
  + Numbers (0-9).
  + Special characters (e.g., !@#$%^&\*()).

**2. Password Generation**

* Randomly generate a password using the selected character sets.
* Ensure the password length matches the user’s input.

**3. Output**

* Display the generated password to the user.

**4. Optional Enhancements (Bonus)**

You can extend the assignment with the following features:

* **Save the Password:** Allow the user to save the generated password to a file.
* **Password Strength Check:** Evaluate the strength of the password and display it (e.g., weak, moderate, strong).
* **User-Specific Preferences:** Allow the user to exclude specific characters or specify mandatory characters.
* **Regenerate Option:** Give the user the option to generate a new password without restarting the program.

**Example Output**

**Input Prompt:**

Enter the desired password length: 12

Include uppercase letters? (yes/no): yes

Include lowercase letters? (yes/no): yes

Include numbers? (yes/no): yes

Include special characters? (yes/no): no

**Generated Password Output:**

Your password is: A7bKc2Lm9dNq

**Hints and Tips**

1. **Generating Random Characters:**
   * Use a randomization library in your programming language to pick characters.
   * Create pools of characters for each type (e.g., a string or list of uppercase letters).
2. **Password Composition:**
   * Combine selected character pools based on user preferences.
   * Randomly pick characters from the combined pool until the password length is met.
3. **Validation:**
   * Ensure at least one character type is selected.
   * If no types are selected, prompt the user to try again.
4. **Code Organization:**
   * Break the program into smaller functions:
     + Input validation.
     + Password generation logic.
     + Output display.

**Deliverables**

Submit the following:

1. The complete source code of your password generator on github.
2. Notes on any bonus features you implemented.