Our objectives are the following:

• Write a single function called random Password to generate random strings.

– The password must be at least 8 characters long. – The password must contain at least 1 number.

– The password must contain at least 1 special symbol. – How you decide to generate this

Password (and how random they are) is up to you beyond these requirements.

• Understand Ascii numbers

• Play with some string functions

# Step 1

# Describe what the function should do. The problem description asks you to write a program, not a function. We will write a password-generating function and call it from the program’s main function.

# Let us be more precise about the function. It will generate a password with a given number of characters.

# We could include multiple digits and special characters, but for simplicity, we decide to include just one of each. We need to decide which special characters are valid. For our solution,

We will use the following set: + - \* / ? ! @ # $ % & the remaining characters of the password are letters.

For simplicity, we will use only lowercase letters in the English alphabet

Step.2

Determine the function’s “inputs”. There is just one parameter: the length of the password.

At this point, we have enough information to document the function: /\*\* Generates a random password.

# At this point, we have enough information to document the function: /\*\* Generates a random password.

# @ Parama length the length of the password @return a password of the given length with one digit and one special symbol \*/

# Step 3

# Determine the types of the parameter variables and the return value.

# The parameter is an integer. The function returns the password, that is, a string.

# The function will be declared as string make password (int length).

Step 4

Write pseudocode for obtaining the desired result. Here is one approach for making a password: Make an empty string called password.

Randomly generate length - 2 letters and append them to password.

Randomly generate a digit and insert it at a random location in password. Randomly generate a symbol and insert it at a random location in password.

How do we generate a random letter, digit, or symbol?

In the spirit of stepwise refinement, we will delegate those tasks to helper functions.

Each of those functions starts a new sequence of steps, which, for greater clarity, we will place after the steps for this function.

Step 5

Implement the function body.

We need to know the “black box” descriptions of the two helper functions described in Step 4 (which we will complete after this function).

Here they are: /\*\* Returns a string containing one character randomly chosen from a given string. @param characters the string from which to randomly choose a character @return a substring of length 1, taken at a random index \*/ string random character(string characters) /\*\* Inserts one string into another at a random position.

@param Str the string into which another string is inserted @param to insert the string to be inserted @return the result of inserting to\_insert into str \*/ string

insert\_at\_random(string str, string to\_insert) Now we can translate the pseudocode of Step 4 into C++: string

make\_password(int length) { string password = ""; for (int i = 0; i < length - 2; i++) { password = password + random\_character("abcdefghijklmnopqrstuvwxyz"); } string random\_digit = random\_character("0123456789"); password = insert\_at\_random(password, random\_digit); string random\_symbol = random\_character("+-\*/?!@#$%&"); password = insert\_at\_random(password, random\_symbol); return password;

Step 6

Test your function.

Because our function depends on several helper functions, we must implement the helper functions first, as described in the following sections.

(If you are impatient, you can use the technique of stubs that is described in Programming Tip 5.5.)

Here,

is a simple program that calls the make\_password function: int main() { srand(time(0)); string result = make\_password(8); cout << result << endl;

return 0;

{

{