A CODING ASSIGNMENT USING C-STRING PROCESSING AND REPETITION STRUCTURE

The Situation:

You have been asked to code a program in C++ that was developed by an Internet Service Provider (ISP) to help them automate the task of assigning account names and passwords to their customers. The ISP has already done the analysis for this task and prepared standard documentation for it below.

Problem Statement:

Request and store a customer's first name, last name, and social security number (SSN). Both names may be entered in any case letters. Allow for names up to 15 characters long. The SSN must be entered using only digits (without hyphens). Generate an account name by concatenating the last four digits of the SSN to the end of the first four characters of the customer's last name in lowercase. Generate a password by concatenating the first four characters (in lowercase) of the customer's first name to the end of the first five digits of the SSN. For example, a customer named "Janet Wilson" with a SSN of "123456789" would be assigned an account name of "wils6789" and a password of "12345jane". For customers with names shorter than four characters, use just the letters provided (understanding that the resulting account name or password may be shorter than standard). The program must display the messages and results as shown on the sample softcopy below.

Sample Softcopy:

The numerals on the left of the sample output below are there for the analyst's reference only and will not appear on the screen. Underlined items indicate values entered by the user.

```
1 Account Generating Program.<CR>
2 Written by Sam Student - 3/3/2011<CR>
3 <CR>
4 First Name? Janet<CR>
5 Last Name? Wilson<CR>
6 Social Security # (digits only)? 123456789<CR>
7 <CR>
8 Username: wils6789<CR>
9 Password: 12345jane<CR>
```

Symbolic Constant List:

Identifier	Description	Data Type	Value	Process Usage	Destination
NSIZE	Maximum size of input names	Integer	15	Index limit for FIRST & LAST	---
SSNSIZE	Maximum size of Soc. Sec. No.	Integer	9	Index limit for SSN	---
USIZE	Maximum size of Username	Integer	8	Index limit for UNAME	---
PSIZE	Maximum size of Password	Integer	9	Index limit for PWORD	---

Variable List:

Identifier	Description	Data Type	Source	Process Usage	Destination
FIRST	Customer's First Name	Char. array	Keyboard	Tested for loop exit and for PWORD	---
LAST	Customer's Last Name	Char. array	Keyboard	for UNAME	---
SSN	Customer's Social Security Number	Char. array	Keyboard	for UNAME and PWORD	---
UNAME	Customer's Account Name	Char. array	Calculated	---	Screen
PWORD	Customer's Password	Char. array	Calculated	---	Screen
PS	Character position in a source string	Integer	Set to 0	String index	---
PT	Character position in a target string	Integer	Set to 0	String index	---

Algorithm:

You might find it easier to follow the approach described in the outline below if you visualize the arrays with some sample data stored in them as shown below.

FIRST	j	a	n	e	t	\0	?	?	?	?	?	?	?	?	?	?
Element:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

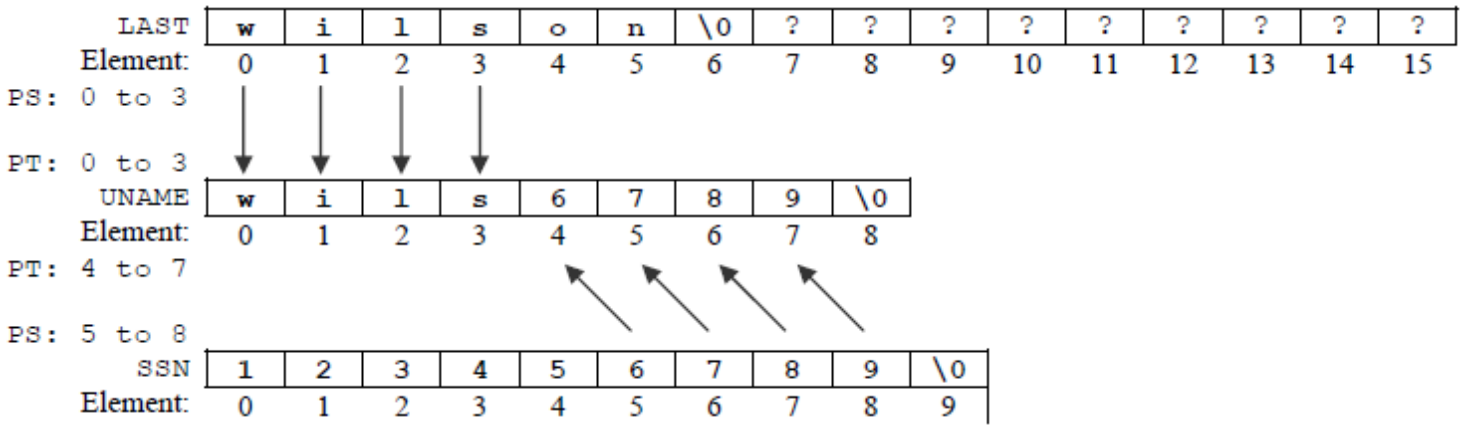
LAST	w	i	l	s	o	n	\0	?	?	?	?	?	?	?	?	?
Element:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

SSN	1	2	3	4	5	6	7	8	9	\0
Element:	0	1	2	3	4	5	6	7	8	9

UNAME	w	i	l	s	6	7	8	9	\0
Element:	0	1	2	3	4	5	6	7	8

PWORD	1	2	3	4	5	j	a	n	e	\0
Element:	0	1	2	3	4	5	6	7	8	9

The scalar variables PS and PT will be used to hold counter values that will point at character element (subscript) positions within the arrays. PS will be used to index the elements in the source arrays, FIRST, LAST, and SSN. PT will be used to index the elements in the resulting arrays, UNAME and PWORD. For example, in step D.1. in the outline, PT is initialized to zero in preparation for indexing (pointing through) the first four elements of UNAME and PS is initialized to zero to index the first four elements of LAST. Next a loop is run through four passes (counting with PT from 0 to 3) to copy the individual characters from the first four elements in the LAST array to the first four elements in the UNAME array. Both PS and PT must be incremented during each pass to keep the array indexes in sync. Then a second loop is run through four passes (counting with PS from 5 to 8) to copy the individual characters from the first four elements in the LAST array to the first four elements in the UNAME array. Both PS and PT must be incremented during each pass to keep the array indexes in sync.



Although programs involving arrays and repetition structures are often developed using flowcharts, an outline style algorithm is provided below for the benefit of students who have not yet learned to use flowcharts well. Pay careful attention to the indentation level of each step in the outline. It will help you to see where to use braces in the C++ source code to block statements so as to nest the structures within the algorithm properly.

Algorithm Outline:

```
A. Start.
B. Display the program title and credits as in the Sample Softcopy.
B.1. Program Title on first line followed by <CR>.
B.2. Credits on second line followed by two <CR>'s.
C. Request and store valid user input from the keyboard.
   C.1. Request and store FIRST.
        C.1.a. Prompt for FIRST as shown on Sample Softcopy Line 4.
        C.1.b. Store keyboard string input in array FIRST.
        C.1.c. Convert any uppercase letters in FIRST to lowercase.
               (Hint: use a counting loop with PS as the counter and the
                character functions isupper and tolower.)
   C.2. Request and store LAST.
        C.2.a. Prompt for LAST as shown on Sample Softcopy Line 5.
        C.2.b. Store keyboard string input in array LAST.
        C.2.c. Convert any uppercase letters in LAST to lowercase.
   C.3. Request and store SSN.
        C.3.a. Prompt for SSN as shown on Sample Softcopy Line 6.
        C.3.b. Store keyboard string input in array SSN.
D. Construct the Username and Password.
   D.1. Build Username from the first 4 characters of the last name,
        followed by the last 4 characters of the SSN.
        D.1.a. Initialize the index for UNAME (Assign zero to PT).
        D.1.b. Copy the first 4 characters of LAST to UNAME.
               (Use a counting loop with PS and PT to copy characters
                from the proper positions in LAST to those in UNAME.
                Be careful to account for short last names.)
        D.1.c. Append last 4 characters of SSN to UNAME.
               (Use a counting loop with PS and PT to copy characters
                from the proper positions in SSN to those in UNAME.)
```

```
        D.1.d. Store an end-of-string after Username (use PT from above).
   D.2. Build Password from the first 5 characters of the SSN,
        followed by the first 4 characters of the first name.
        D.2.a. Initialize the index for PWORD (Assign zero to PT).
        D.2.b. Copy the first 5 characters of SSN to PWORD.
               (Use a counting loop with PS and PT to copy characters from
                the proper positions in SSN to those in PWORD.)
        D.2.c. Append first 4 characters of FIRST to PWORD.
               (Use a counting loop with PS and PT to copy characters from
                the proper positions in FIRST to those in PWORD. Be careful
                to account for short last names.)
        D.2.d. Store an end-of-string after Password (use PT from above).
E. Display and identify Username and Password.
   E.1. Skip a line (#7 in Sample Softcopy) on the display screen.
   E.2. Display and identify UNAME as shown on Sample Softcopy Line 8.
   E.3. Display and identify PWORD as shown on Sample Softcopy Line 9.
F. End
```

Desk Check:

The desk check for this algorithm has been omitted for the sake of brevity in this assignment. Desk checks for programs that have arrays manipulated by loops can be quite long. So some analysts take a shortcut by starting a check of the loop by going through the first few passes and then jump to a value of the control variable near the final value of the loop and finishing the test there. This practice has obvious risks, but is widely practiced by many professionals (with varied rates of success).