EECE 230 –Test No. 2

Date: December 6, 2010

Three Problems (Problems 1 and 2, each worth 30 points; Problem 3 worth 40 points):

Problem 1

Define three arrays of characters, each of which has 16 characters. The first one is called `fNM`, the second one is `mNM`, and the third one is `lNM`. Write a program that asks the user to enter his or her first name and store it in `fNM`. Then ask the user to enter his or her middle name and store it in `mNM`, and then his or her last name and store it in `lNM`. In each case you need to make sure that the entered string can fit in the arrays.

Use a function that you pass it all three names and make it build an electronic signature and return it to main, where you will display it.

The signature is constructed as follows: first letter of `fNM`, followed by the first letter of `mNM`, followed by the first letter of `lNM`, followed by the second letter of `fNM`, followed by the second letter of `mNM`, followed by the second letter of `lNM`, and so on until `span` letters from each name is taken. The value of `span` is equal to the length of the shortest name passed to the function. Next, append to this constructed string the digits of the sum of the two longest names.

To convert an integer variable to a string (array of characters), use the built-in function `itoa(int val, char buffer[], int base)`. Below is an example of how to use it.

```
#include <string>
int counter = 33;
char Name[16]="Mustapha", buffer[8];
itoa(counter, buffer, 10); //the value 10 at the end is fixed and indicates that counter is decimal
strcat(name, buffer);
cout<<"Name is: "<<name<<endl; //the output will be Mustapha33
```

Example

Please enter your first name: Christopher
Please enter your middle name: Imad
Please enter your last name: Karkafi

The electronic signature of your name is: CIKhmararidk18

In the above, "Imad" is the shortest name, and has four letters, so we take four letters from each name; one letter at a time from each name. Next, we sum the lengths of the longest two names (in this case the length of the first and the length of the last), which is equal to 11+7=18.

Problem 2

Declare an array of characters of size 65 and call it `sentence`. Ask the user to enter a sentence of up to 64 characters, which could include spaces.

Ask the user to select one of five options:

1. Capitalize every word in the sentence
2. Delete a specified word from the sentence (if it is found)
3. Switch to lower case every word in the sentence, except the first word.
4. Move the last word in the sentence to the start of the sentence
5. Exit the program

Example

- Please enter a sentence: The Sheppard who cried wolf and lied to the village
- Now, consider the following options:
  1. Capitalize every word in the sentence
  2. Delete a specified word from the sentence (if it is found)
  3. Switch to lower case every word in the sentence, except the first word.
  4. Move the last word in the sentence to the start of the sentence, meaning that you should insert it at the beginning of the sentence
  5. Exit the program

- Please select one of the five options above: 1
- The sentence is now: The Sheppard Who Cried Wolf And Lied To The Village
- Please select another one of the five options above: 2
- Please specify the word to remove: who
- The sentence is now: The Sheppard Cried Wolf And Lied To The Village
- Please select another one of the five options above: 4
- The sentence is now: Village The Sheppard Cried Wolf And Lied To The
- Please select another one of the five options above: 5

Problem 3

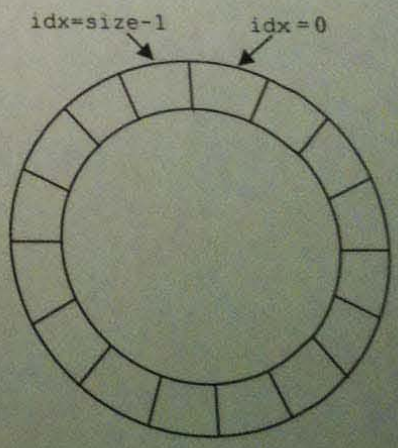
Suppose you have an array and you are supposed to write a function that inserts values in this array in duplicate and in a circular fashion, as is explained below. This function returns a `bool` and takes as input four parameters:

1. The integer array `iCircAr`
2. The size of the array `size`
3. The *positive* integer value `posVal` to be inserted in duplicate
4. The index of the slot (`Idx`) in `iCircAr` where to insert `posVal`

- This function first checks if the array slot whose index is `Idx` is free (i.e., no positive value is stored in it).
  - If it is free, it inserts `posVal` at that slot *and* also inserts it at `size/2` slots away.
  - If this function finds that this slot is not free, it checks the next slot, i.e., at index `Idx+1`. If this slot is free, it inserts `posVal` at that slot *and* at `size/2` slots away.
  - If this function finds that slot `Idx+1` is not free, it checks the next slot: at index `Idx+2`. If this slot is free, it inserts `posVal` at that slot *and* at `size/2` slots away.
  - The above procedure repeats until the function finds an empty slot, or determines that the array is full.
  - If the function is able to insert `posVal`, it returns true, otherwise it returns false.
  - Note that if the index of the slot being examined is `size-1`, and that this slot is full, the function should next check if slot whose index is 0 to see if it is empty.

Effectively, the array is treated like a circular array whose end is connected to its start, as shown in the figure below.

In `main()`,declare an integer array of size 20 and allow the user to enter positive integer values and their places in the array until the array is full. Every time the user supplies a value and an index, call the function to insert the value. When the function returns a false (meaning the array is full), print the content of the array and exit.



Example

- The user enters 13 and 7. The function inserts the value 13 in slot **7** and in slot 7+10=**17**.
- The user enters 44 and 0. The function inserts the value 44 in slot **0** and in slot 0+10=**10**.
- The user enters 25 and 7. The function inserts the value 25 in slot **8** (slot 7 was full) and in slot 8+10=**18**.
- The user enters 1 and 10. The function inserts the value 1 in slot **11** (10 was full) and in slot 11+10=21⇒ slot **1** (use modulo 20).
- The user enters 70 and 17. The function inserts the value 70 in slot **19** (17 and 18 were full) and in slot 19+10=29⇒ slot **9** (use modulo 20).