CMPE – 211
Preliminary Work (Pre-Lab Activity)
Laboratory Experiment #3

Textbook Material:	Chapters 5-6	pp.87-155	[see Laboratory Experiments #2]
	Chapter 7	«Pointers and References»	pp.156-182
	Chapter 8	«C-Strings»	pp.183-196

• • • TASK 1

Write, Compile and Execute a C++ program that enters a **string** from the user (a full stop `.` indicates the end of the input string) and **finds and displays the vowel characters** ('a', 'e', 'i', 'u', 'o') contained in it (both upper- and lowercase characters are processed). The program includes the following function prototypes:

```
void read_str(char str[ ], int& n);          // enters the characters from the user
int display_vowels(char str[ ], int n);      // returns the number of vowels
```

Example: For the input string "`This is the next lab experiment`", the program outputs:

ieeaeeie

• **NOTE:** For this and all following tasks You are free to use any basic constructions and data types (*scalar*, i.e. integral, floating-point, pointers; *compound*, e.g. enumerations, arrays) of C++ the most appropriate for the implementation. We assume that a core of these topics is already known from previous C language-based courses – the task uses so-called C-string(s).

• • • TASK 2

Write, Compile and Execute a C++ program that enters a **string** from the user (a full stop `.` indicates the end of the input string), **removes all vowels** and **prints** out the resulting string. The program includes the following function prototypes:

```
void remove_vowels(char str[ ], int& n);     // removes the punctuation marks
void print_str(char str[ ], int n = 0);      // prints out the resulting string
```

Example: for the input string "`Hi? How's the life?`", the program outputs: *H? Hw's th lf?*

• • • TASK 3

Many computerized check-writing systems do not print the amount of the check in words. Perhaps the main reason for this omission is the fact that most high-level computer languages used in commercial applications do not contain adequate string manipulation features. Another reason is that the logic for writing word equivalents of check amounts is somewhat involved.

Write, Compile and Execute a C++ program that inputs a numeric check amount and writes the word equivalent of the amount. For example, the amount 2112 should be written as

TWO THOUSAND ONE HUNDRED TWELVE

The initial part of the program may look as follows:

```
#include <iostream>
using namespace std;
int main( )
{
     const char *digits[10] = { " ", "ONE", "TWO", "THREE", "FOUR", "FIVE", "SIX",
                                "SEVEN", "EIGHT", "NINE" };
     const char *teens[10] = { " ", "ELEVEN", "TWELVE", "THIRTEEN", "FOURTEEN",
                               "FIFTEEN", "SIXTEEN", "SEVENTEEN", "EIGHTEEN", "NINETEEN" };
     const char *hundred = "HUNDRED";
     const char * thousand = "THOUSAND";
     const char *tens[10] = { " ", "TEN", "TWENTY", "THIRTY", "FOURTY", "FIFTY", "SIXTY",
                              "SEVENTY", "EIGHTY", "NINETY" };
     int yeni_TL;
```

- continued next page –

- **HINT:** Most of calculations in the program are based on division / and modulus division % operators

Hypothetical dialogue may have the form given below:

```
Enter the check amount (0 to 9999):  2555
The check amount in words is:
TWO THOUSAND FIVE HUNDRED and FIFTY FIVE YTL
```

• • • TASK 4

Write, Compile and Execute a C++ program that performs encryption/decryption operations. User-defined function **encrypt( )** takes a character pointer as a parameter and uses array-subscript notation to change values in the character array by **adding 1 (one)** to each entry. Function **decrypt( )** takes a character pointer as a parameter and uses pointer notation to change values in the character array by **subtracting 1 (one)** from each entry. Function **main( )** calls functions **encrypt( )** and **decrypt( )** and prints the encrypted string.

The initial part of the program may look as follows:

```
#include <iostream>
using std :: cout;
using std :: endl;

void encrypt(char [ ]);        // prototypes of functions in use
void decrypt(char * ePtr);
int main( )
{
        // create a string to encrypt
        char string[ ] = "this is a secret!";
        cout << "Original string is: " << string << endl;
        encrypt( string );
        // call to the function encrypt( )
        cout << "Encrypted string is: " << string << endl;
        decrypt( string );
        // call to the function decrypt( )
        cout << "Decrypted string is: " << string << endl;
        return 0;
}  ...  ...
```

Example:

```
Original string is: this is a secret!
Encrypted string is: uijt!jt!b!tfdsfu"
Decrypted string is: this is a secret!
```

• • • Appendix

- Check Review Questions at the end of textbook's Chapters 5-8 (pp. *112*, *142*, *172* and *200*) and review pointer arithmetic rules and C-string processing

• • • Sources

- *John R.Hubbard*. Schaum's Outline of Programming with C++, 2nd edition, McGraw-Hill, 422 p., 2000
- *Harvey M.Deitel, Paul J.Deitel*. C++ How To Program, 4th edition, Prentice Hall, 1320 p., 2002
- *Harvey M.Deitel, Paul J.Deitel*. C++ in the Lab (Lab Manual to Accompany C++ How To Program, 4th edition), 629 p., 2003