National University of Computer and Emerging Sciences

Assignment 01

Object Oriented Programming

Course Instructor(s)	Rizwan Ul Haq
Semester	Spring 2025
Open Date	07-Feb-2024
Submission	14-Feb-2024

FAST School of Computing

Submission Instruction

1. Make a folder by the name in the following format. Assignment_01_RollNumber (Assignment_01_23F1234). **Copy all .cpp files only** (not the whole project) with name of the question (Q1.cpp, Q2.cpp) in this folder. Compress it and submit on google classroom.
2. A .docx file containing all the codes and screenshots should also be included.
3. The codes must be properly commented.
4. Assignment will be marked as zero in case of any plagiarism. Submit your own work only.
5. **Any assignment not following the above given instructions will not be evaluated at all.**

Question#1	**5**

Write a program that asks the user to enter two doubles as inputs to be stored in the variables var1 and var2 respectively. There are also two double pointers named ptrV1 and ptrV2. Point the values of' var1 and var2 by ptrV1 and ptrV2 respectively, and display those using the pointers.

Question#2	**10**

FASTlish is a modified form of English Language developed by the developers of FAST NU. The main purpose of this language is to communicate with a fellow Fastian in an encrypted form. In this language basically the letters of English language are replaced by some other letters. There is a one-to-one mapping. So for example every English letter has mapped letter. For example ‘a’ might be mapped to ‘y’ and ‘o’ is mapped to ‘e’.

There is an underlying scheme of every letter mapped to some other letter. The secret recipe to generate a word in this language lies in the roll no. of a student. For example one student has a Roll no. 18F-WXYZ. Now there are 2 secret codes for the encoded characters one code is W+X+1 for the capital Letters and second code is Y+Z+1 for the small letters.

Part 1:

Now write a Program in which you declare 2 integers a=W+X+1 and b=Y+Z+1. Now create 2 Pointers, P1 will point to a and P2 will point to b.

Then write a function **void GenerateCode(int* P1,int* P2,char* line)**. This function takes a line by reference and changes it

into the encrypted line.

Example scenario:

My Roll no is 19F-0805. So Code 1 is 0+8+1 = 9 and Code 2 is 0+5+1 = 6.

Now For Capital letters Code is 9 and For Small letters Code is 6.

Input: Hello World!

Output: Qkrru Fuxrj!

So every capital letter is replaced by (W+X+1)th character ahead and every small letter is replaced by (Y+Z+1)th character ahead. Hint: W is replaced by F. So assume after ‘z’ there comes an ‘a’ again and after ‘Z’ there comes an ‘A’.

Part 2:

After Encoding the characters, you need to decode the character and do everything above exactly in the opposite manner. Your Decode function will be like
void Decode (int* P1, int* P2, char* line).
Main function:
your main function should be something like this:
…
Print Original Line
Enter roll number
GenerateCode(....)
Print Encoded Line
Decode(...)
Print Decoded Line
…
Console Output:
Console Screen of your output should be exactly like this:
Original Line: Hello World!
Encoded Line: Qkrru Fuxrj! (You might have a different encoded line)
Decoded Line: Hello World! (Must be same as original line)

Question#3	**10**

CString library provides us different function some of them are listed below. 20
1. **strlen** - get string length
int strlen (const char *s);
2. **strcmp** - compare two strings
int strcmp(const char *s1, const char *s2);
3. **strncmp** - compare parts of two strings
int strcmp(const char *s1, const char *s2,int n);
4. **strcpy** - copy a string
char *strcpy(char * s1, const char *s2);
5. **strncpy** - copy part of a string
char * strcpy(char * s1, const char *s2, int n);
6. **strcat** - concatenate two strings
char *strcat(char * s1, const char * s2);
7. **strncat** - concatenate one string with part of another
char *strcat(char * s1, const char * s2,int n);
You are required to write own definition of these functions that will result in the same functionality as the CString library functions do. You have to implement all the required checks yourself.
Note: You cannot use any built-in library to perform these string operations.

Question# 4	**5**

Suppose you have a main() with three local arrays, all the same size and type (say float). The first two are already initialized to values. Write a function called addarrays() that accepts the addresses of the three arrays as arguments; adds the contents of the first two arrays together, element by element; and places the results in the third array before returning. A fourth argument to this function can carry the size of the arrays. Use pointer notation throughout; the only place you need brackets "[]" is in defining the arrays

Question# 5	**15**

Write a program that takes two matrices from user and performs following operations:

- Matrix Addition
- Transpose of a matrix
- Checks if a matrix is symmetric or not
- Interchange rows of a matrix

You are supposed to implement following functions:
Note: You cannot use array subscript operator[] to traverse the array. Use only pointer notation.

1. **double** ReadMatrix(const char *filename, int& rows, int& cols)**
   Description: This function will take size of matrix from file, create a matrix dynamically, take matrix elements from file and return the matrix created.
2. **void OutputMatrix(double** matrix, const int& ROWS, const int& COLS)**
   Description: Displays the matrix in proper format.
3. **double** AddMatrix(double** matrixA, double** matrixB, const int& ROWS, const int& COLS)**
   Description: This function takes two matrices as parameters, adds them and saves the result in a newly created matrix R and returns the result.
4. **double** TransposeMatrix(double** matrix, int& ROWS, int& COLS)**
   Decription: This function takes a matrix A, takes transpose of matrix A, saves the result in a newly created matrix and returns the result.
5. **bool IsSymmetric(double** matrix, const int& ROWS, const int& COLS)**
   Description: This function takes a matrix as parameter with its size information and returns true if the matrix is symmetric and false otherwise.
6. **void InterchangeRows(double** matrix, const int& ROWS, const int& COLS)**
   Description: This function takes two row numbers from the user and calls following function to actually interchange the rows.
7. **void InterchangeRows(int*& row1, int*& row2)**
   Description: This function interchanges two rows. **You are NOT ALLOWED to iterate through rows and swap their values.** Think of simple solution.

Important Notes:

- You cannot change the prototypes of the functions.

- You can use subscript operator to allocate and deallocate the memory.
- Your program should follow the exact sequence of Sample Run given below.
- Goto instruction is not allowed in your program.
- Violation of any of the above instructions may result in ZERO credit or marks deduction.

Sample Run (with sample inputs):

```
Matrix A =
1.2     2.3     3.4
4.3     5.0     6.7
7.2     8.8     9.6

Matrix B =
2.3     5.2     8.4
5.2     6.8     9.1
8.4     9.1     10.7

Matrix C =
2.2     3.5     4.1
5.3     6.4     7.8

A + B =
3.5     7.5     11.8
9.5     11.8    15.8
15.6    17.9    20.3

A+C =
Addition not possible.

Transpose of A =
1.2     4.3     7.2
2.3     5.0     8.8
3.4     6.7     9.6

Transpose of C =
2.2     5.3
3.5     6.4
4.1     7.8

Matrix A is NOT Symmetric

Matrix B is Symmetric

Interchanging Rows of Matrix A:
```

```
row1: 1
row2: 3

After Interchanging Rows Matrix A=
7.2     8.8     9.6
4.3     5.0     6.7
1.2     2.3     3.4
```

Note: These are only sample inputs. User can enter any value supported by data type.

Q6	15

(Simulation: The Tortoise and the Hare) In this exercise, you'll re-create the classic race of the tortoise and the hare. You'll use random number generation to develop a simulation of this memorable event.

Our contenders begin the race at "square 1" of 80 squares. Each square represents a possible position along the race course. The finish line is at square 80. The first contender to reach or pass square 80 is rewarded with a pail of fresh carrots and lettuce. The course weaves its way up the side of a slippery mountain, so occasionally the contenders lose ground.

There is a clock that ticks once per second. With each tick of the clock, your program should use function moveTortoise and moveHare to adjust the position of the animals according to the rules in Table 1. These functions should use pointer-based pass-by-reference to modify the position of the tortoise and the hare.

Table 1: Rules of the game

Animal	Move Type	Percentage of Time	Actual Move
Tortoise	Fast plod	50%	4 squares to right
	Slip	20%	5 squares to left
	Slow plod	30%	1 square to right
Hare	Sleep	38%	No move at all
	Big hop	20%	11 squares to right
	Big slip	20%	9 squares to left
	Small hop	22%	1 square to right

Use variables to keep track of the positions of the animals (i.e., position numbers are 1–80). Start each animal at position 1 (i.e., the "starting gate"). If an animal slips left before square 1, move the animal back to square 1.

Generate the percentages in Table 1 by producing a random integer i in the range $1 \leq i \leq 50$. For the tortoise, perform a "fast plod" when $1 \leq i \leq 25$, a "slip" when $26 \leq i \leq 35$ or a "slow plod" when $36 \leq i \leq 50$. Use a similar technique to move the hare.

For each tick of the clock (i.e., each iteration of a loop), display an 80-position line showing the letter T in the tortoise's position and the letter H in the hare's position. Occasionally, the contenders land on the same square. In this case, the tortoise bites the hare and your program should display

OUCH!!! beginning at that position. All positions other than the T, the H or the OUCH!!! (in case of a tie) should be blank.

After displaying each line, test whether either animal has reached or passed square 80. If so, display the winner and terminate the simulation. If the tortoise wins, display TORTOISE WINS!!! YAY!!! If the hare wins, display Hare wins. Yuck. If both animals win on the same clock tick, you may want to favor the tortoise (the "underdog"), or you may want to display It's a tie. If neither animal wins, perform the loop again to simulate the next tick of the clock.

HAPPY CODING