ASSIGNMENT 1

Course: Object Oriented Programming

Teacher: Sir Zulqurnain

Section: BS(AI)2A,2B

Instructions:

- Do not share your code with anyone else. All assignments are to be done individually.
- **Plagiarized Solutions** of any type will lead to ZERO marks.
- Submit only your source code `.cpp` files. Name your file as rollno_name_dept_sec.c (e.g., `P22-1234_Name_AI_1A.c`).
- Follow the naming scheme correctly; failure will result in ZERO marks.
- Submissions only through respective **Google Classroom** will be entertained.
- Make sure your code compiles and runs; non-compiling code will get ZERO marks.
- You are only allowed to use pointers not index allowed and you cant use strings use only char arrays(c strings).
- Late submissions will not be entertained.
- If there is confusion, ask the TA or make reasonable assumptions, but only those that do not conflict with the question.
- Happy Coding

Question 1 : Word processor for FAST

A rapidly growing digital publishing company has decided to build an intelligent **Text Processing Engine** to automate routine editing tasks. This engine will be used by editors, writers, and content reviewers who work with articles, books, blogs, and academic documents on a daily basis.

Due to the high volume of text being processed, the company needs a fast and reliable system that can manipulate textual data efficiently using low-level memory operations. You have been hired as a junior software engineer to develop the core module of this system.

Your task is to design a **menu-driven C++ program** that allows users to interactively apply different text operations through a single interface. To ensure compatibility with other parts of the system, you must strictly follow the given function prototypes.

System Constraints:

- The program must be written in **C++**.
- Character arrays and pointers must be used for string manipulation.
- Function prototypes must not be changed.
- Dynamic memory allocation should be used wherever required.

User Interaction Menu

When the program runs, it should repeatedly display the following menu until the user chooses to exit:

```
========== TEXT PROCESSING ENGINE ==========
1. Merge two strings
2. Compress a string
3. Reverse a sentence
4. Convert singular words to plural
5. Exit
Enter your choice:
```

Each menu option corresponds to a specific feature of the Text Processing Engine described below.

Option 1: String Merging Utility

Editors often need to join fragments of text coming from different sources. This option allows the user to input two strings and merge them into one.

The contents of the first string must be appended to the **end of the second string** using the following function:

```
void StringConcatenate(char *str1, char *str2);
```

Constraint: No additional string or temporary buffer is allowed inside the function.

Option 2: Text Compression Tool

To reduce redundancy and save storage space, the system must eliminate duplicate characters from words while keeping the order of their first appearance.

Implement the following function to perform this operation:

```
void CompressString(char *);
```

Examples:

- `"abadca"` → `"abdc"`
- `"aaaaaaabbaad"` → `"abd"`

No extra array or string may be used during this process.

Option 3: Sentence Reversal Feature

Writers sometimes want to experiment with sentence structure. This option allows the user to reverse the order of words in a sentence without altering the original sentence.

The function must return a newly created reversed sentence:

```
char* ReverseSentence(char *);
```

Example:

- Input: `"I am Pakistani"`
- Output: `"Pakistani am I"`

Option 4: Automated Grammar Assistant

The Text Processing Engine also includes a grammar assistant that converts singular nouns into their plural forms based on predefined linguistic rules.

Implement the following function:

```
void pluralWords(char **s, int wordCount);
```

The function receives a dynamically allocated 2D array of words and must update each word to its plural form by reallocating memory when necessary.

Pluralization Rules

1. Append `es` if:
    - The word ends with `h` and the second last letter is `c` or `s`
    - The word ends with `x` or `is`
    - The word ends with `s` or `z` and the second last letter is not a vowel
    - The word ends with `ato`
2. Append `zes` if the word ends with `z` and the second last letter is a vowel (`a` or `e`)
3. Append `ses` if the word ends with `s` and the second last letter is a vowel (`a` or `e`)
4. If the word ends with `ff`, append `s`
5. If the word ends with `f` or `fe`, replace it with `ve` and append `s`
6. If the word ends with `on`, replace `on` with `a`
7. If the word ends with `us`, replace `us` with `i`
8. If the word ends with `y`:
    - If the second last letter is a vowel, append `s`
    - Otherwise, replace `y` with `ies`
9. For all other cases, append `s`

Question 2: New Flex

Due to the rapidly increasing cost of RAM, the university administration has ordered all in-memory systems to be redesigned for **maximum memory efficiency**. One such system is the **University Marks Management System**, which processes student assessments in real time.

As an initial optimization step, only **quiz marks** are being handled.

New Quiz Storage Policy

For a class with N students:

- Student 1 stores quiz marks of quizes 1 to N
- Student 2 stores quiz marks of quizes 2 to N
- Student 3 stores quiz marks of quizes 3 to N
- Student N stores quiz marks for 1 quiz only

This creates a **jagged structure** that avoids unused memory allocation and ensures memory efficiency.

Input Requirements

Your program must ask the user for:

1. Total number of classes
2. Number of students in each class
3. Quiz marks for each student according to the jagged policy:
    - For student 1 $\rightarrow$ enter N quiz marks
    - For student 2 $\rightarrow$ enter $N - 1$ quiz marks
    - ...
    - For student N $\rightarrow$ enter 1 quiz mark

    All memory for storing quiz marks must be dynamically allocated using pointers only.

Data Structure Requirements

- Dimension 1 $\rightarrow$ Classes
- Dimension 2 $\rightarrow$ Students within each class
- Dimension 3 $\rightarrow$ Quiz marks stored according to the policy

No indexing, STL, or built-in functions are allowed.

Menu

```
========== QUIZ ANALYSIS SYSTEM ==========
1. Enter quiz marks
2. Display quiz marks
3. Calculate student average
4. Calculate class average
5. Find maximum quiz mark in a class
6. Identify best performing class
7. Adjust quiz marks for fairness
8. Exit
Enter your choice:
```

Functional Requirements

1. Enter Quiz Marks

Allow the user to enter quiz marks for each class and each student following the jagged structure.

2. Display Quiz Marks

Display all stored quiz marks class-wise and student-wise, clearly showing the jagged nature of the data.

3. Calculate Average Marks of a Student

The user should select:

- Class number
- Student number within that class

The program computes the average of all quizzes available for that student:

$$StudentAverage = \frac{Sumofquizzesforstudent}{Q_i}$$

where Q_i = number of quizzes stored for that student.

—

4. Calculate Average Marks of a Class

The user should select the **class number**.

Compute the average of each student individually, then compute the average of these student averages:

$$ClassAverage = \frac{\sum(StudentAverages)}{TotalStudentsintheclass}$$

This ensures fairness despite unequal quiz counts.

—

5. Find Maximum Quiz Mark in a Class

The user selects the class number, and the program identifies the highest quiz mark in that class by traversing all stored quizzes.

6. Identify Best Performing Class

Compare class averages and determine the class with the highest average. If multiple classes share the same average, report all of them.

7. Adjust Quiz Marks for Fairness

Since students attempt different numbers of quizzes, direct comparison is unfair. Scale each quiz mark using:

$$AdjustedMark = OriginalMark \times \frac{N}{Q_i}$$

where:

- N = total number of students in the class
- Q_i = number of quizzes stored for student i

After adjustment:

- Recalculate student averages
- Recalculate class averages

—

Memory Management Rules

- Only dynamic memory allocation using pointers is allowed
- No indexing ([]) is allowed
- No STL, no built-in containers or string functions
- All allocated memory must be properly deallocated before program termination

Example User Interaction

```
========== QUIZ ANALYSIS SYSTEM ==========
1. Enter quiz marks
2. Display quiz marks
3. Calculate student average
4. Calculate class average
5. Find maximum quiz mark in a class
6. Identify best performing class
7. Adjust quiz marks for fairness
8. Exit
Enter your choice: 1

Enter number of classes: 2

Enter number of students in class 1: 3
Enter quiz marks for student 1 (3 marks): 80 90 85
```

```
Enter quiz marks for student 2 (2 marks): 75 95
Enter quiz marks for student 3 (1 mark): 88

Enter number of students in class 2: 2
Enter quiz marks for student 1 (2 marks): 90 92
Enter quiz marks for student 2 (1 mark): 85

========== QUIZ ANALYSIS SYSTEM ==========
Enter your choice: 2
Displaying quiz marks:

Class 1:
Student 1: 80 90 85
Student 2: 75 95
Student 3: 88

Class 2:
Student 1: 90 92
Student 2: 85

========== QUIZ ANALYSIS SYSTEM ==========
Enter your choice: 3
Calculate student average:
Enter class number: 1
Enter student number: 2
Student 2 in Class 1 Average: 85.0

========== QUIZ ANALYSIS SYSTEM ==========
Enter your choice: 4
Calculate class average:
Enter class number: 1
Class 1 Average: 84.67

========== QUIZ ANALYSIS SYSTEM ==========
Enter your choice: 5
Find maximum quiz mark in a class:
Enter class number: 2
Maximum quiz mark in Class 2: 92

========== QUIZ ANALYSIS SYSTEM ==========
Enter your choice: 6
Identify best performing class:
Best performing class: Class 2 (Average: 89.0)

========== QUIZ ANALYSIS SYSTEM ==========
```

```
Enter your choice: 7
Adjust quiz marks for fairness:
Quiz marks have been adjusted using the formula:
Adjusted Mark = Original Mark * (Total Students in Class / Number of Quizzes for Student

========== QUIZ ANALYSIS SYSTEM ==========
Enter your choice: 2
Displaying adjusted quiz marks:

Class 1:
Student 1: 80.0 90.0 85.0
Student 2: 112.5 142.5
Student 3: 264.0

Class 2:
Student 1: 90.0 92.0
Student 2: 170.0

========== QUIZ ANALYSIS SYSTEM ==========
Enter your choice: 8
Exiting program...
```

Made with love by human Hint: on next page