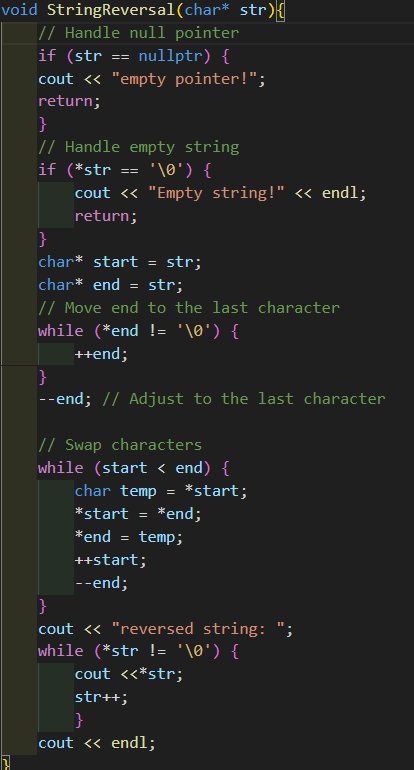
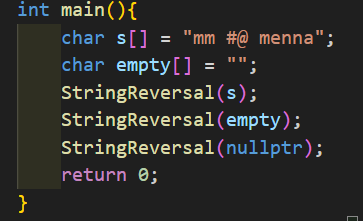
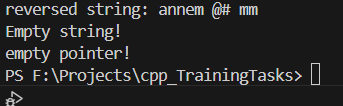
**Problem 11 (1) string reversal**

StringReversal() employs two pointers (start and end) to swap characters from the beginning and end of the string towards the center until they meet. And handle nullptr case if true return.



Output:



Return the reversed string for s

And warning for empty pointers and strings.

A diagram of a program

Description automatically generatedFlow chart:

**Problem (13) sparse matrix using linked list**

Sparse matrix is a matrix with a majority of zero elements. using a linked list, we implement functions to perform matrix addition with another sparse matrix.

**A computer screen shot of a black screen

Description automatically generatedFirst step:** made class node to make linked lists node with

1. row-> to store the row of non-zero elements
2. col-> to store the columns of non-zero elements
3. value-> to store the data of this element
4. \*next-> pointer to the next node

**A screenshot of a computer program

Description automatically generatedSecond step:**

Create nodes take parameters:

1. Node pointer to pointer (Head for linked list)
2. Row index for non-zero data
3. Col index for non-zero data
4. Value of the data

We make temp point to the head of linked list to traverse the linked list later

if temp is nullptr then we create the first node else we add new node at the end of the linked list

A screen shot of a computer code

Description automatically generated

**Third step:**

The function iterates through the 2D array and creates a linked list node for each non-zero element in the matrix.

Parameters:

1. Node \*\*head: double pointer to the head of the linked list
2. int \*\*matrix: A pointer to a 2D array
3. int rows: The number of rows in the 2D array.
4. int cols: The number of columns in the 2D array.

A screenshot of a computer program

Description automatically generated**Forth step:**

addSparseMatrices() adds two sparse matrices represented as linked lists and returns the result as a new linked list.

Parameters:

list1 and list2 representing the two sparse matrices.

Then we initialized three pointers of type Node

1. Node \*result = nullptr;  
   is a pointer that will point to the head of the resulting linked list.
2. Node \*ptr1 = list1;  
   is a pointer used to traverse the first input linked list
3. Node \*ptr2 = list2;

is a pointer used to traverse the second input linked list

Then loop Through Both Lists:

1. **Case 1: Same Position in Both Matrices:** 
   1. the values at this position are added.
   2. If the sum is non-zero, a new node with this sum is added to the result linked list.
   3. Both ptr1 and ptr2 are moved to their respective next nodes.
2. **Case 2: ptr1 Comes Before ptr2:**
   1. ptr1 value is added to the result list.
   2. ptr1 is moved to the next node.
3. **Case 3: ptr2 Comes Before ptr1:**
   1. ptr2 value is added to the result list.
   2. ptr2 is moved to the next node.

**Appending Remaining Nodes:**

After the main loop, there may still be remaining nodes in either list1 or list2 that haven't been added to the result list. (When sparse matrix that has non-zero elements in rows and columns that the another does not.)

Two while loops handle this:

1. The first loop adds all remaining nodes from list1 to result.
2. The second loop adds all remaining nodes from list2 to result.

**Test case:**A screen shot of a computer program

Description automatically generated

Create two sparse matrices, represent them as linked lists, and then add these matrices using the addSparseMatrices()

1. Define the dimensions of the 2D arrays(4,5). The sparseMatrix pointer will store the addresses of the rows, and each row pointer will point to an array of integers representing the non-zero elements in that row.
2. Creating and Initializing the First Sparse Matrix. sparseMatrix is dynamically allocated with 4 rows and 5 columns, initialized with zeroes. Specific non-zero values are assigned to sparseMatrix at certain positions
3. Converting the First Matrix to a Linked List
4. Do the same with sparseMatrix\_2
5. Then we add the two linked lists using addSparseMatrices()
6. printing the linked lists, deallocating the memory for the 2D arrays and linked lists

A screenshot of a computer program

Description automatically generated**A screen shot of a computer program

Description automatically generatedResults:**

A screenshot of a computer

Description automatically generated