CS221-L Data Structures and Algorithms Lab

Lab report # 5

Submitted by: Shayan Rizwan

Registration Number: 2024585

Submitted to: Adnan Haider

Semester: 3rd

Faculty of Computer Science and Engineering

GIK Institute of Engineering Sciences and Technology

Task 1: Reverse a String Using a Stack

Write a C++ function to reverse a string using a stack, excluding spaces.

Requirements:

- Take a string as input.
- Reverse all non-space characters using a stack.
- Ignore spaces in the reversed output.

Example:

- Input: "Hello World"
- Output: "dlroWolleH"

Learning Objective: Practice using stack push and pop operations to reverse sequences.

Solution

CODE:

```
#include <iostream>

using namespace std;

// NAME: Shayan Rizwan Yazdanie
// REG #: 2024585

void ReverseStrings(string& str) {
  const int cap = 100;    // max stack size
  char stack[cap];         // manual stack implemented as an array
  int top = -1;                 // stack pointer, -1 means empty stack

  // Push non-space characters onto the stack
  for (int i = 0; i < (int)str.size(); i++) {
    if (str[i] != ' ') {
      if (top < cap - 1) {   // check for overflow
        stack[++top] = str[i];  // push character onto stack
      }
```

```
    }
  }

 // Now top + 1 is the number of non-space chars

 // Since top is the index of the top element, and indexing is zero-based:
// Number of elements in the stack = top + 1.

  int newLength = top + 1;

  // Resize string to the number of non-space chars
  str.resize(newLength);

  // Pop characters from stack and assign to string indices

  // Because the stack stores characters in the original order, popping them
(last-in-first-out) gives them in reversed order.
  for (int i = 0; i < newLength; i++) {
    str[i] = stack[top--];
  }

  // Using top-- (postfix decrement):
  // Reads the character at current top.
  // Then decrements top.
}


int main() {

  string test;
  cout << "Enter a string: ";
  getline(cin, test);
  ReverseStrings(test);
  cout << test << endl;  // Output: dlrowolleh




  return 0;
}
```

OUTPUT

```
Enter a string: Hello there, this will be a reversed string!
!gnirtsdesreveraeblliwsiht,erehtolleH
```

Task 2: Balanced Parentheses Checker

Write a C++ function to check if a string of parentheses is balanced using a stack.

Requirements:

•Return True if every '(' has a matching ')' in correct order (also for "{}" and "[]").
•Return False otherwise.

Example:

•Input: "(())()"
•Output: True (expression is balanced) / False (expression is not balanced)

Learning Objective: Understand how stacks help validate nested structures through matching pairs.

Solution

CODE:

```
#include <iostream>

using namespace std;

// NAME: Shayan Rizwan Yazdanie
// REG #: 2024585

bool isBalanced(const string& s) {
```

```
  char stack[1000];  // stack array
  int top = -1;    // stack pointer

  for (int i = 0; i < (int)s.length(); i++) {
    if (s[i] == '(') {
      top++;
      stack[top] = '(';  // push '('
    } else if (s[i] == ')') {
      if (top == -1) {
        return false;  // no matching '('
      }
      top--;  // pop '('
    }
  }

  return (top == -1);  // true if all '(' matched
}

int main() {

  string user_string = "(";

  if (isBalanced(user_string)) {
    cout << "TRUE" << endl;
  }
  else {
    cout << "FALSE" << endl;
  }

  return 0;
}
```

Task 3: DJ Music Playlist – Undo Last Played Songs (Using Stack)

Imagine you are a DJ controlling a live playlist. The stack represents your play order, where the last song added is the first to play. Simulate this using stack operations implemented manually with arrays (no STL).

Menu Options:

- Add Song => Push
- Play Song => Peek
- Delete Song => Pop
- Show Playlist => Display/Show
- Exit

Expected Example:

------- DJ Music Playlist -------

1. Add Song
2. Play Song
3. Delete Song
4. Show Playlist
5. Exit

Enter choice: 1
Enter Song Name: Breath of Life
Song "Breath of Life" added!

Enter choice: 1
Enter Song Name: EDM_Night
Song "EDM_Night" added!

Enter choice: 4
Playlist: [Breath of Life, EDM_Night]

Enter choice: 2
Playing: EDM_Night

Enter choice: 3
Song "EDM_Night" deleted!

Enter choice: 5
Exit…. (DJ ended)

Learning Objective: Understand stack operations through real-life entertainment.
Use case: LIFO/FILO logic using a DJ playlist system.

Solution

CODE:

```
#include <iostream>

using namespace std;

// NAME: Shayan Rizwan Yazdanie
// REG: 2024585

void addSong(string songs[], int &top, int size) {
  if (top == size - 1) {
    cout << "Playlist is full!\n";
    return;
  }
  cout << "Enter Song Name: ";
  cin.ignore();
  getline(cin, songs[++top]);
  cout << "Song \"" << songs[top] << "\" added!\n";
}

void playSong(string songs[], int top) {
  if (top == -1) {
    cout << "No songs to play!\n";
    return;
  }
  cout << "Playing: " << songs[top] << endl;
}
```

```
void deleteSong(string songs[], int &top) {
  if (top == -1) {
    cout << "No songs to delete!\n";
    return;
  }
  cout << "Song \"" << songs[top--] << "\" deleted!\n";
}

void showPlaylist(string songs[], int top) {
  if (top == -1) {
    cout << "Playlist is empty!\n";
    return;
  }
  cout << "Playlist: [";
  for (int i = 0; i <= top; i++) {
    cout << songs[i];
    if (i < top) cout << ", ";
  }
  cout << "]\n";
}

int main() {
  const int size = 50;  // flexible limit, not a macro
  string songs[size];
  int top = -1;
  int choice;

  cout << "------- DJ Music Playlist -------\n";
  cout << "1. Add Song\n2. Play Song\n3. Delete Song\n4. Show Playlist\n5.
Exit\n";

  do {
    cout << "\nEnter choice: ";
    cin >> choice;

    switch (choice) {
      case 1:
        addSong(songs, top, size);
        break;
      case 2:
        playSong(songs, top);
        break;
      case 3:
```

```
        deleteSong(songs, top);
        break;
      case 4:
        showPlaylist(songs, top);
        break;
      case 5:
        cout << "Exit... (DJ ended)\n";
        break;
      default:
        cout << "Invalid choice!\n";
    }
  } while (choice != 5);

  return 0;
}
```

OUTPUT:

```
------- DJ Music Playlist -------
1. Add Song
2. Play Song
3. Delete Song
4. Show Playlist
5. Exit

Enter choice: 1
Enter Song Name: Breath of Life
Song "Breath of Life" added!

Enter choice: 1
Enter Song Name: EDM_Night
Song "EDM_Night" added!

Enter choice: 1
Enter Song Name: Diamonds
Song "Diamonds" added!

Enter choice: 2
Playing: Diamonds

Enter choice: 3
Song "Diamonds" deleted!

Enter choice: 4
Playlist: [Breath of Life, EDM_Night]

Enter choice:
```