**Command Line Calculator**

**Project Overview 📝**

Command line calculator provides a simple solution for performing calculations through CLI, but it also supports the user input, incase if no argument was passed, user will pass the arguments as string either with-in the quotes or without quotes and program will generate the result. The program supports this format only: "101 - ( 2 + 43 )", 101 – ( 2 + 43 ), or "101 - ( 2 + 43 ) else it will throw an error for invalid expression. User can change values and brackets placements but single quotes, curly braces, power, log are not supported by this program, moreover, the program does not handle double quotes as if you provide it or not then still it will work fine.

Whole program is relying on two classes i.e., Utility and Controller class, check these classes in detail below:

**Utility Class:** This class serve as a helper for our program as it includes following member functions which supports to build logic in Controller class, let’s explore each member function in detail below:

1. **precedence:** This function accepts single char type argument and returns the operators integer type precedence.
2. **check\_brackets\_balanced:** This function is designed to check if brackets in given expression are balanced or not, it accepts single string as argument and return Boolean value.
3. **validate\_operators:**  It takes string as argument and checks whole expression and return false if any invalid input was given else return true if expression given expression pass the test, for example: 101 - ( 2 + 43 ) or "101 - ( 2 + 43 ) "✔️ and 101 – (( 2 + 43 )}) ❌.
4. **fix\_spaces:** A small function is designed to fix the spaces for postfix expression. It takes string and adjust spaces and returns the updated string.
5. **infix\_to\_postfix:** This function is designed to return the string type postfix expression while manipulating the original expression given by user.
6. **evaluate\_postfix\_expression:** Last function of Utility class that returns the final integer type expression sum. It receives a postfix expression and generate the final result.

**Controller Class:** This class is built on top of the Utility class as it utilizes all utility class functions by creating a Utility class object as a private member variable and then it uses utils powers to control the whole application logic. This class includes a single member function as follows:

1. **RunApp:** This is a main entry point function of this program, as it initializes the app. It takes two arguments, one size, an integer type value and second argument, a two-dimensional char array. It first checks the size if it is greater than one or not, if yes than work on given arguments and if “no” then it inputs a expression and evaluate the results and print them to the console.

The main function contains Controller class object, which calls the RunApp method to initialize the program, and hence program go with going.

**­Getting Started 🛠️**

To get started with this project follow these steps:

1. **Clone this repository:**

git clone <https://github.com/saqibbedar/dsa.git>

cd .\dsa\Assignments\02\_Assignment

1. **Compile and run C++ program:**

g++ -o main main.cpp // compile C++ program

.\main.exe // you can pass arguments here i.e., "101 - ( 2 + 43 )" (optional)

1. **Compile and run C++ program in isolated container using Docker (optional):**

If you are docker user then you can build a docker image and run this program in an isolated environment. To run C++ using docker follow the instructions below correctly, the **Dockerfile** is already setup in 02\_Assignment directory, so just run these commands to create docker image and start a container and test the application.

**Step 1: Build docker image:**

docker build -t <docker-image-name> . // replace <docker-image-name> i.e., main-cpp etc.

**Step 2: Run your C++ program:**

docker run --rm -it main-cpp

**Contributions 😍**

If you'd like to contribute:

* You can add your code to the [Contribute](https://github.com/saqibbedar/DSA/tree/main/Contribute) directory.
* For more extensive contributions, please create a separate repository with your organized code.
* If you find any errors in my solutions or have improvements, feel free to suggest updates.
* Please refer to the [CONTRIBUTE.yaml](https://github.com/saqibbedar/dsa/blob/main/Contribute/CONTRIBUTE.yaml) file for detailed contribution guidelines.

**License 📄**

This project is licensed under MIT License – see the [LICENSE](https://github.com/saqibbedar/dsa?tab=MIT-1-ov-file) file for more details.