Stringulation

|Description

A simple assembly program that performs some of the main string operations:

- **InsertAt(Dword , string)**
- **Replace(byte , byte)**
- **Compare(string)**
- **Remove (Dword , Dword)**
- **Find(string)**

The Languages used:

- **Assembly with Irvine Library**

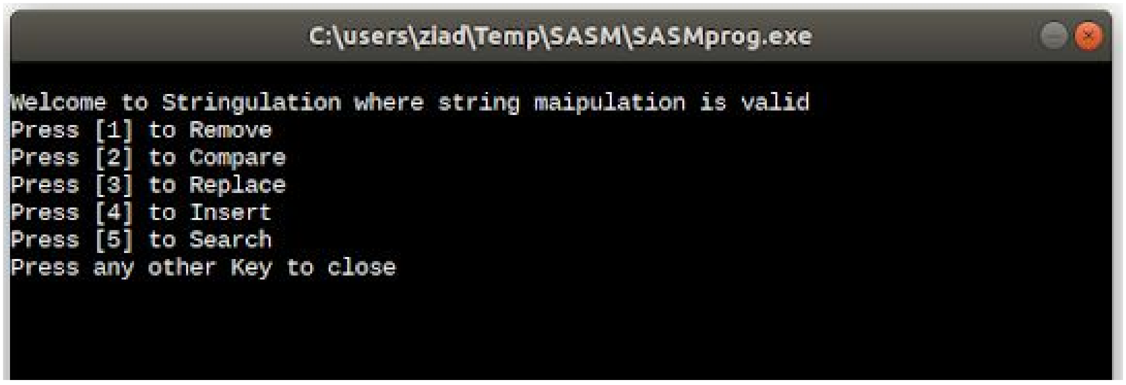
The constraints:

- $0 \leq |MainString| + |NewString| \leq 10^5$, if $NewString$ existed.
- $0 \leq Index \leq |MainString|$
- $0 \leq cnt + Index < |MainString|$

|Main Procedures

- **Main**

  The main contains the implementation of a simple UI for the user. The code runs through an infinite loop and depending on the user input a certain function is called or the program closes.

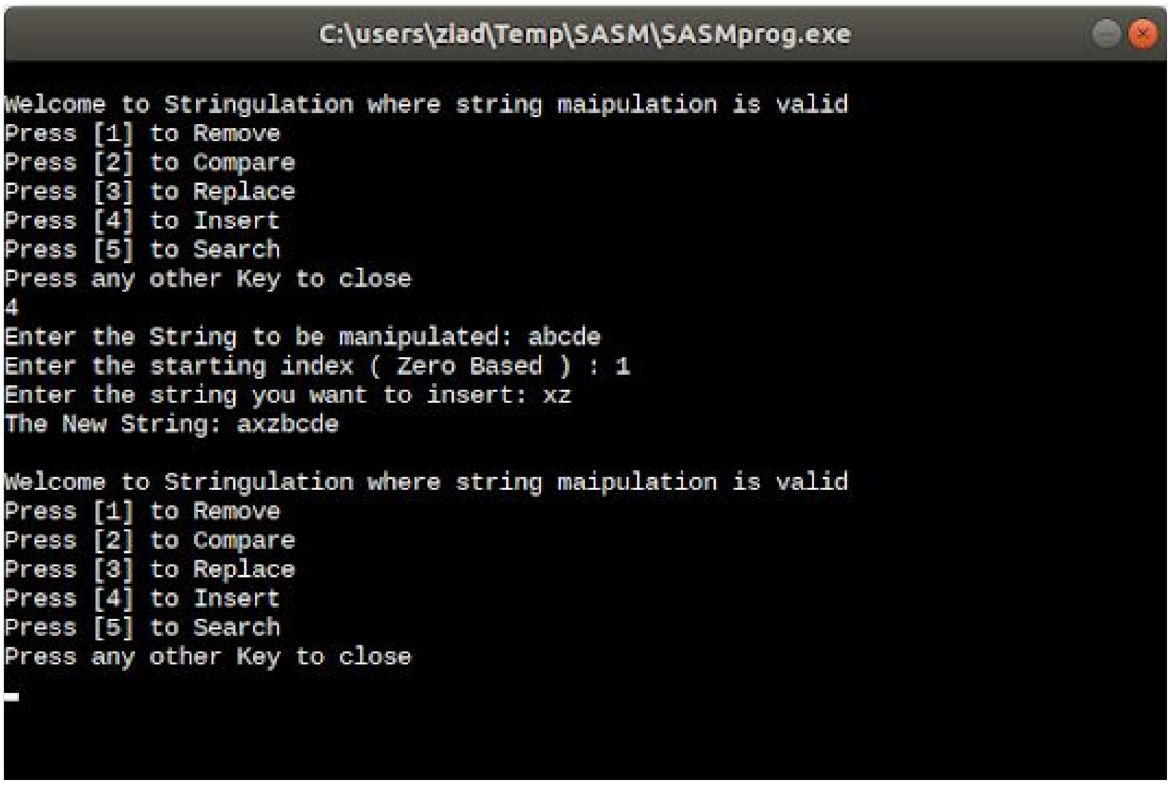


```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
```

- **InsertAt (Dword index , string newString)**

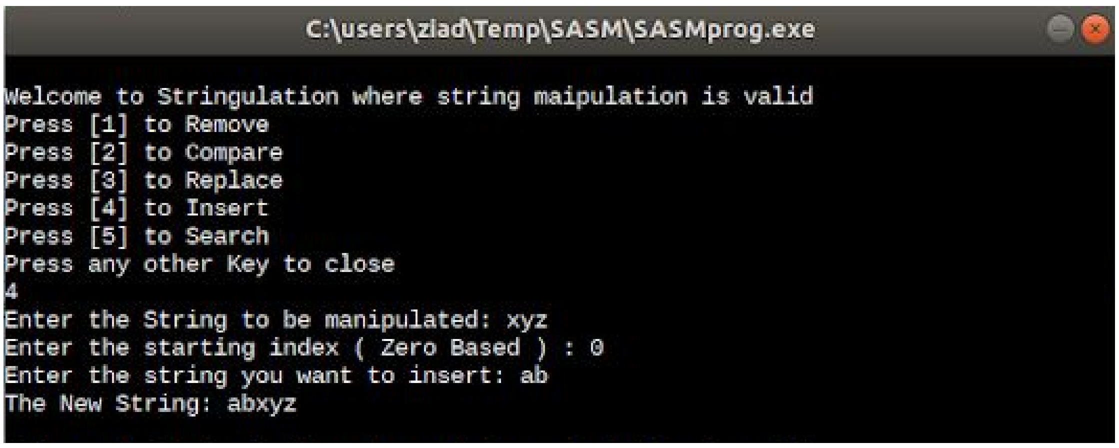
The $index$ is pushed in a stack frame and the $newString$ is a global string. The logic of the function runs as follows. All the characters starting from the index entered by the user (the starting index of the newly added string) are shifted to the right by the length of the new string.
Input = $abcde\ 1\ xz$, the modified string = $abcbcde$
then loop from the starting index for new string size steps and overwrite the main string by the new string. The final string = $axzbcde$



C:\users\ziad\Temp\SASM\SASMprog.exe

```
Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
4
Enter the String to be manipulated: abcde
Enter the starting index ( Zero Based ) : 1
Enter the string you want to insert: xz
The New String: axzbcde

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
```



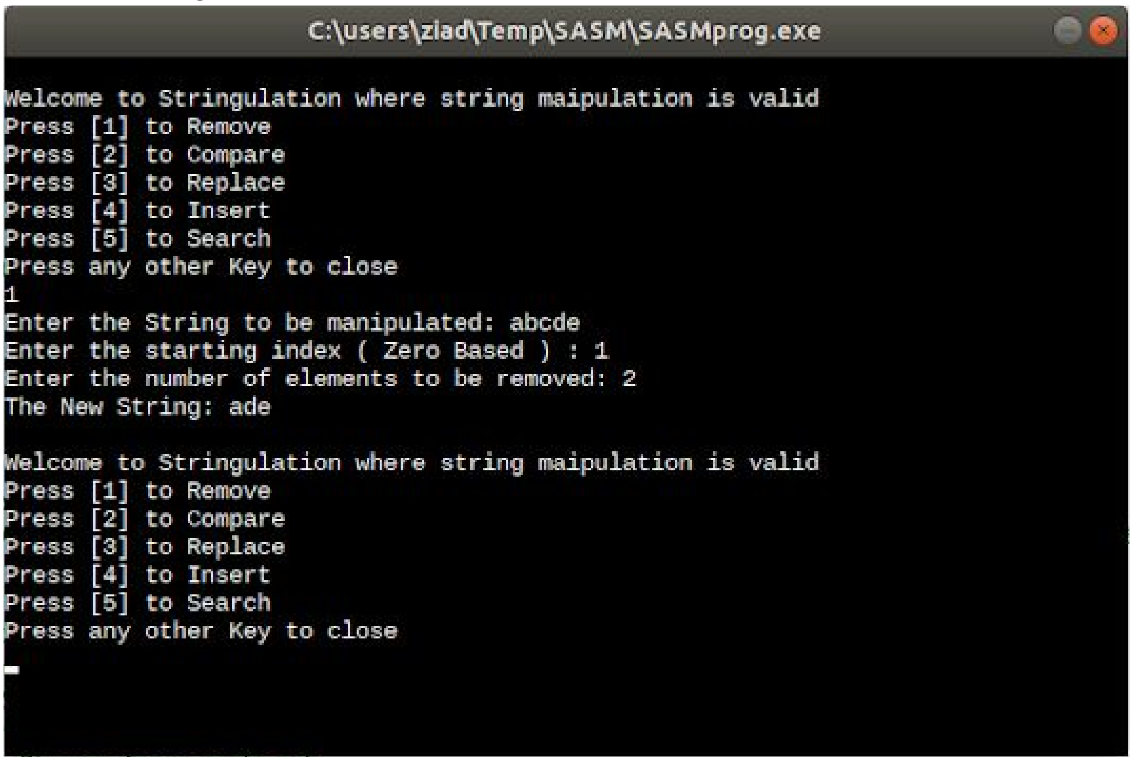
C:\users\ziad\Temp\SASM\SASMprog.exe

```
Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
4
Enter the String to be manipulated: xyz
Enter the starting index ( Zero Based ) : 0
Enter the string you want to insert: ab
The New String: abxyz
```

- **Remove (Dword index, Dword cnt)**

  The $index$ and the cnt are pushed in a stack frame. The remove function removes starting from an $index$ for the length of cnt starting from it. It looks a bit like the insert as a similar technique is used. Two pointers are used, one points to the starting index of the removal and the other one points to the character after the last place to be removed. Then the characters are overwritten.

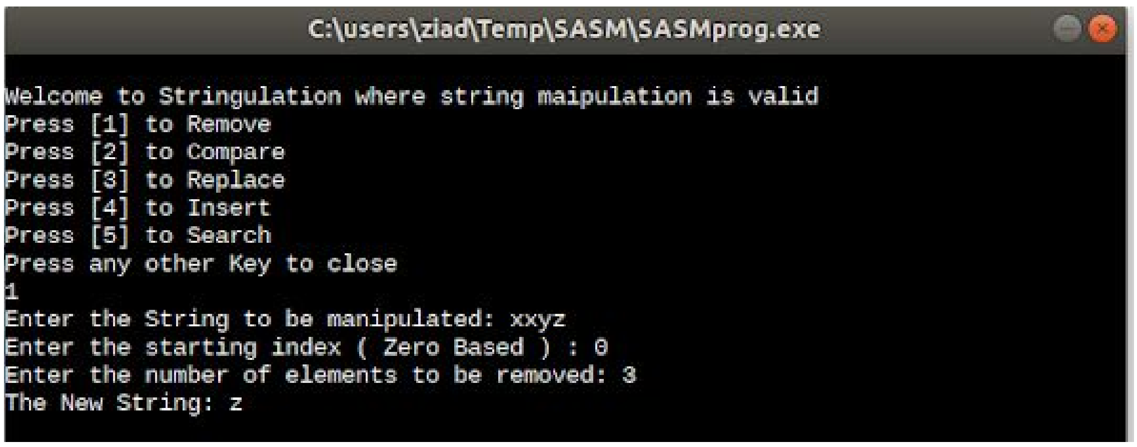
  Input = $abcde\ 1\ 2$, the modified string = $adede$. But a problem still exists because the string still contains extra characters, so a null terminator character is added after the desired string (null terminator index = $|MainString| - |NewString|$) and the main string size is modified. The final string = $ade\backslash 0$ but the null character doesn't appear; it only indicates the end of the string.



```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
1
Enter the String to be manipulated: abcde
Enter the starting index ( Zero Based ) : 1
Enter the number of elements to be removed: 2
The New String: ade

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
```

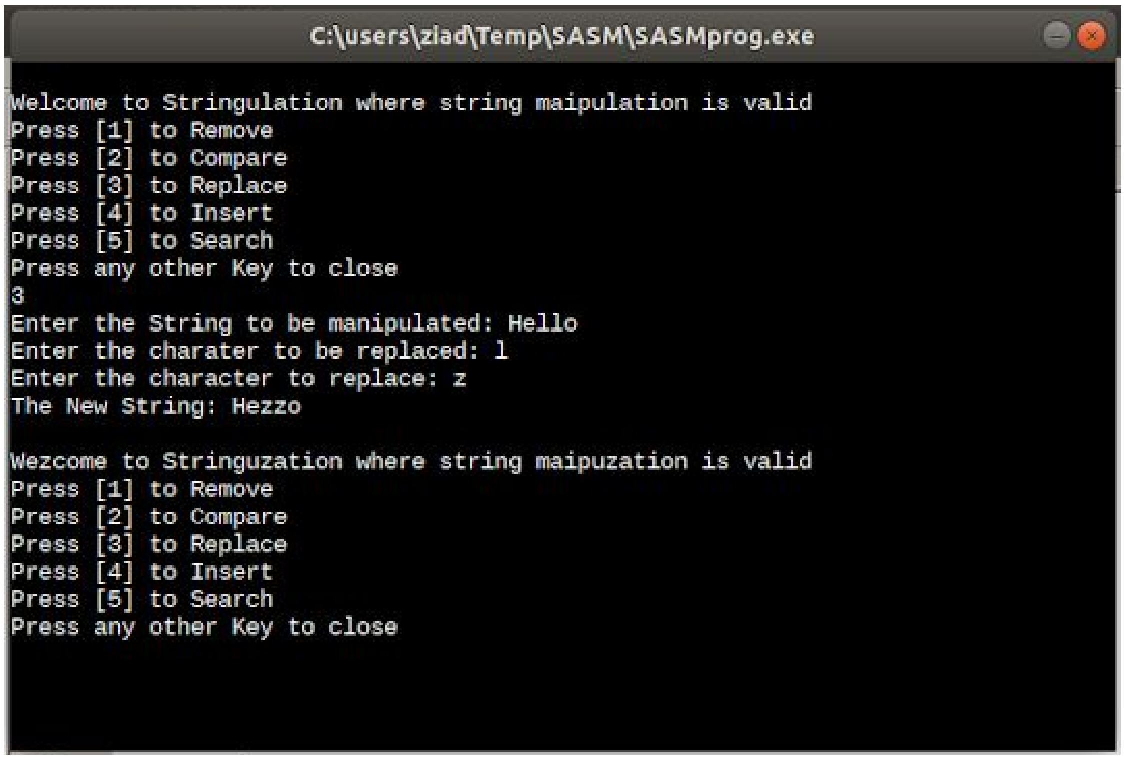


```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
1
Enter the String to be manipulated: xxyz
Enter the starting index ( Zero Based ) : 0
Enter the number of elements to be removed: 3
The New String: z
```

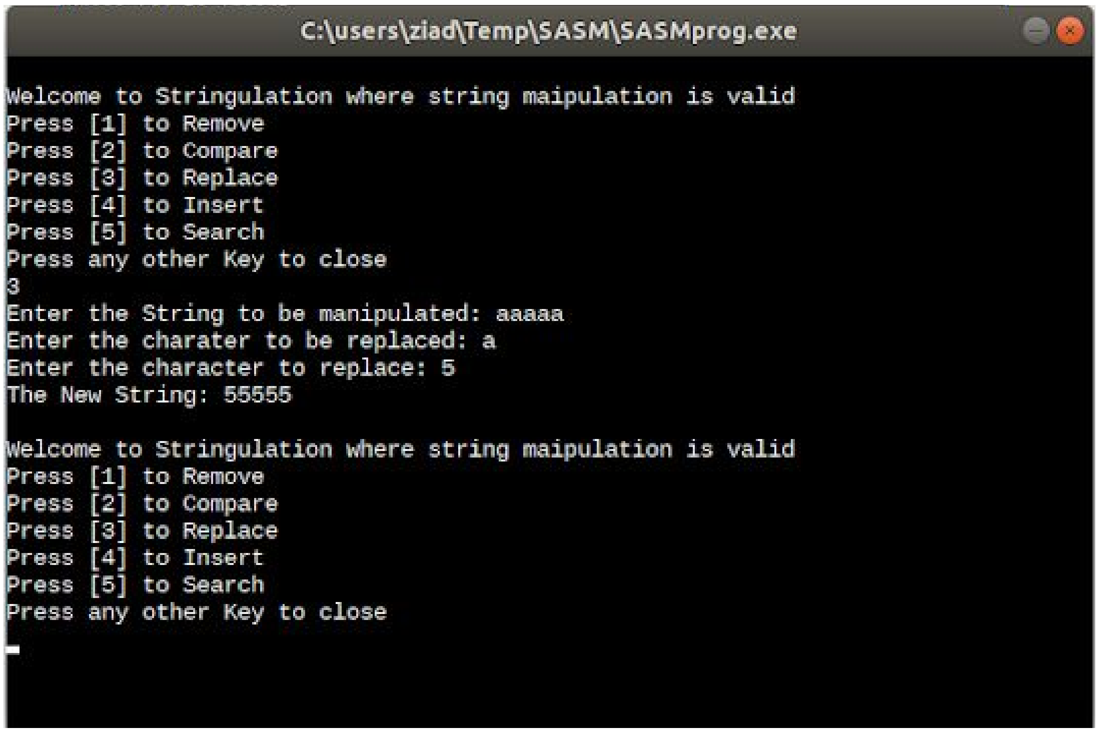
- **Replace (byte ToBeReplaced , byte ToReplace)**

  The two parameters are pushed in a stack frame. The easiest implementation which consists of only a loop that iterates over all the string and whenever the character we want to replace appears it is replaced instantaneously.
  Input = $Hello\ l\ z$
  Output = $Hezzo$



```
Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
3
Enter the String to be manipulated: Hello
Enter the charater to be replaced: l
Enter the character to replace: z
The New String: Hezzo

Wezcome to Stringuzation where string maipuzation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
```

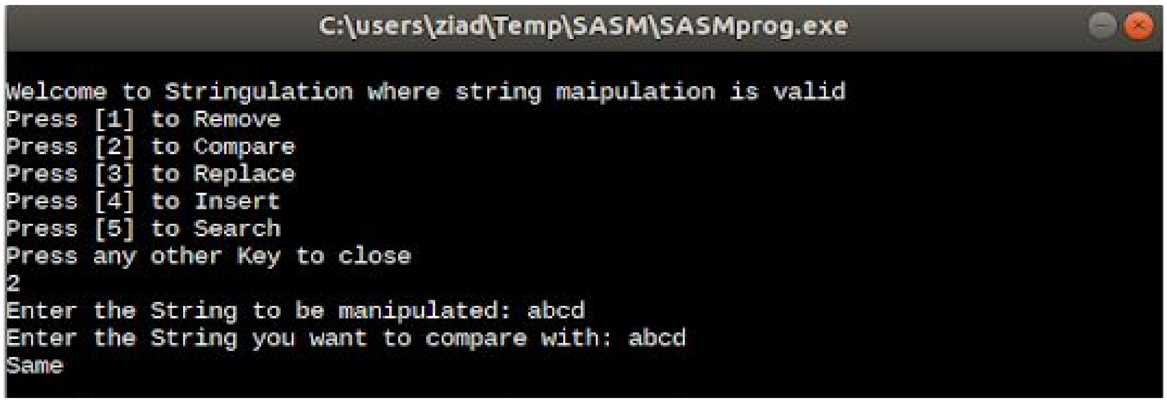


```
Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
3
Enter the String to be manipulated: aaaaa
Enter the charater to be replaced: a
Enter the character to replace: 5
The New String: 55555

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
```

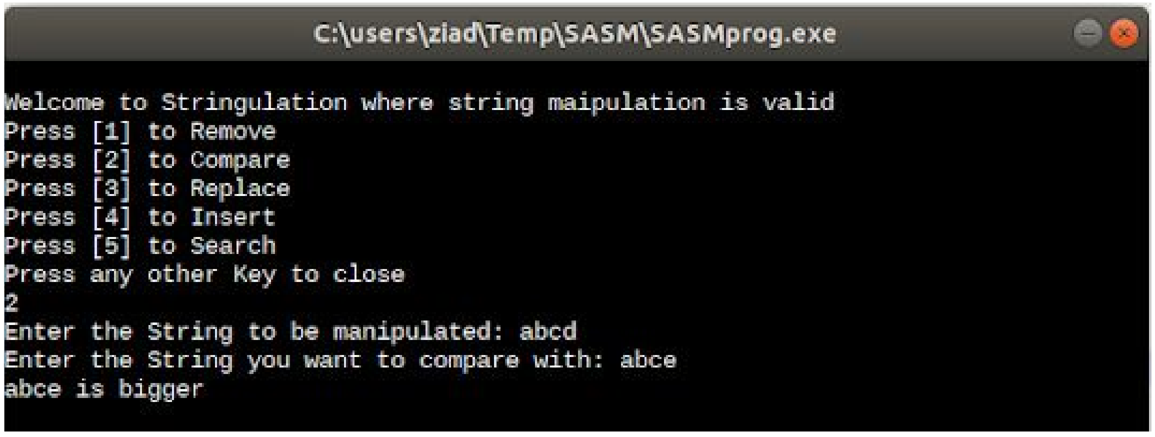
- **Compare (string newString)**

  $NewString$ is a global variable declared in the data section. The function compares between two strings; the main string and the string passed to the function. Then it prints the bigger one lexicographically. A loop iterates for the minimum string size iteration over the two strings using two pointers; each one points to the same index character in the two strings. When the two corresponding characters are not the same, a comparison takes place between the two characters and the bigger string is found. If one of the two strings is a substring of the other one the bigger in size is the bigger lexicographically. If the two strings are the same a message **" Same "** is printed.



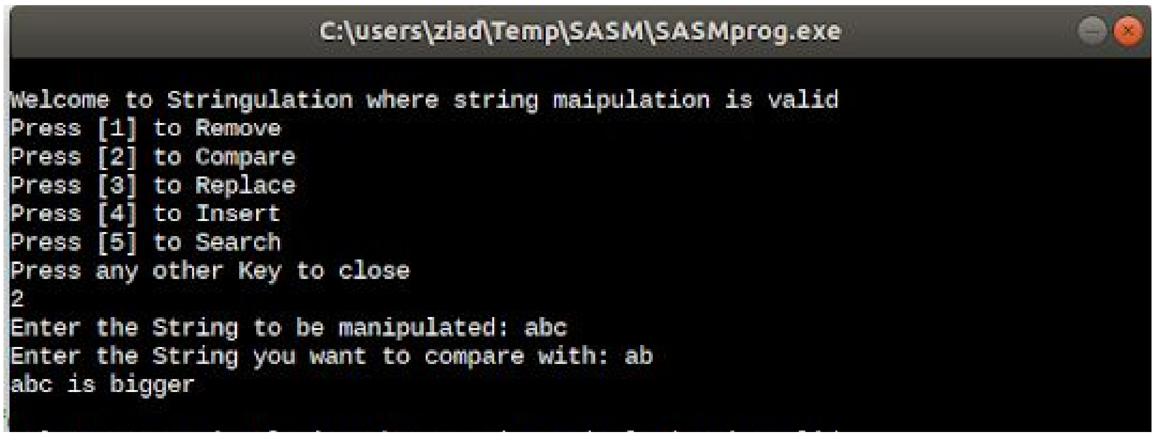
```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
2
Enter the String to be manipulated: abcd
Enter the String you want to compare with: abcd
Same
```



```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
2
Enter the String to be manipulated: abcd
Enter the String you want to compare with: abce
abce is bigger
```

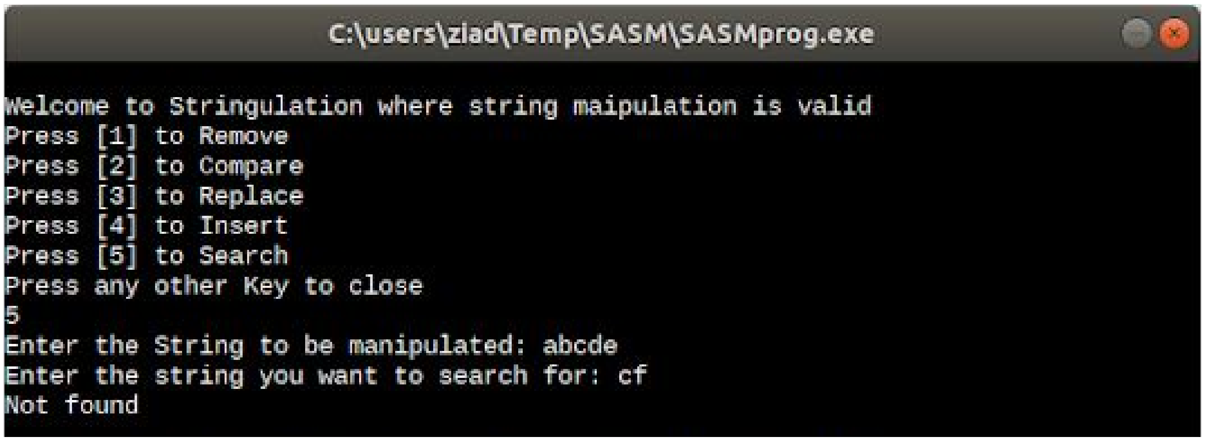


```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
2
Enter the String to be manipulated: abc
Enter the String you want to compare with: ab
abc is bigger
```

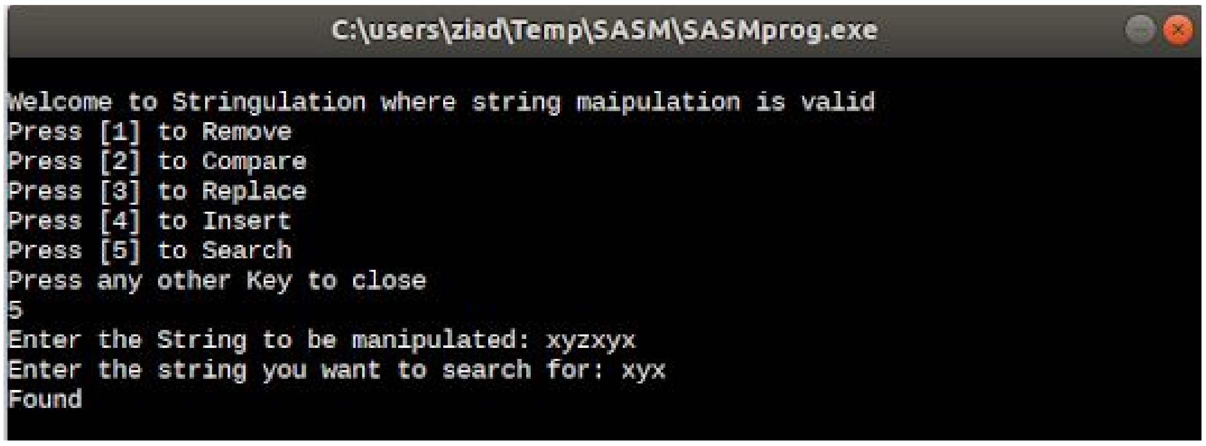
- **Find (String newString)**

  The function takes a string to be searched for in the main string. Its implementation is a little bit challenging. First of all if the $|NewString|$ is greater than the $|MainString|$ "Not found" is printed directly. Nested loops are used where the outer loop iterates a maximum of $|MainString| - |NewString| + 1$ iterations only which checks whether the first character of the new string equals the character where the iterator points to in the main string. If the two characters are equal the inner loop part just steps in the compare each character in the new string the character of the main string using primitive string instruction. Thus, I had to check for the last character again when the loop ends as the string primitive instruction increments on matter what, so if the two characters are equal a message "Found" is printed, else "Not found" is printed.



```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
5
Enter the String to be manipulated: abcde
Enter the string you want to search for: cf
Not found
```

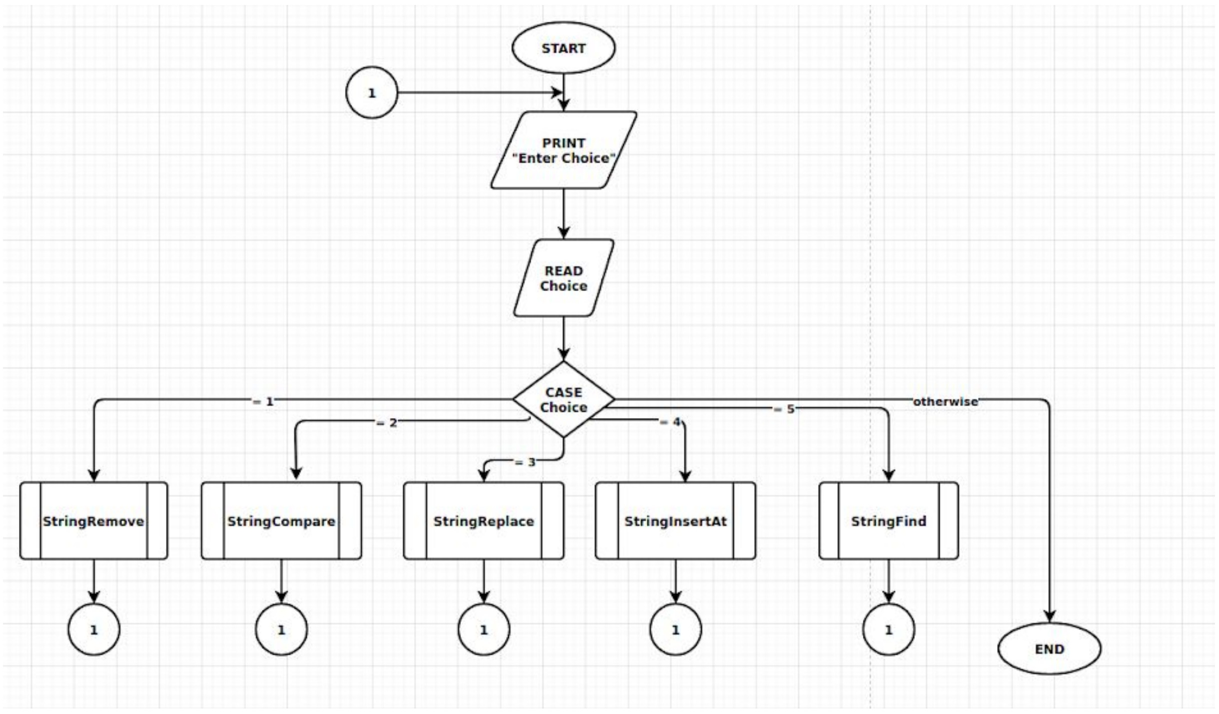


```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
5
Enter the String to be manipulated: xyzxyx
Enter the string you want to search for: xyx
Found
```

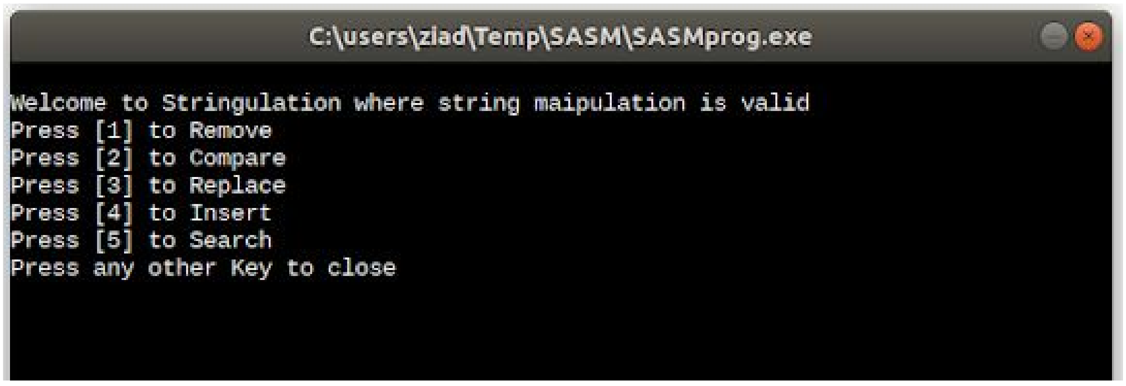
|Flow Chart

The whole program flowchart can be found through this link for a better view.



|UserManual

- After running the program a UI appears



```
C:\users\ziad\Temp\SASM\SASMprog.exe

Welcome to Stringulation where string maipulation is valid
Press [1] to Remove
Press [2] to Compare
Press [3] to Replace
Press [4] to Insert
Press [5] to Search
Press any other Key to close
```

- Choose the string operation you want to perform by entering its number then press enter
  1. **Remove:**
     The main string is entered at first, then the index you want to remove characters starting from it (zero-based), lastly the number of characters you want to remove starting from the entered index.
  2. **Compare:**
     This function compares between two entered strings then it displays the bigger one lexicographically.
  3. **Replace:**
     Replacement of a certain character with another selected character. The main string is entered first, then the character to be replaced, lastly the character to replace it.
  4. **Insert:**
     Insertion of string in another string. First the main string is entered, then the index where you want to insert the other string in, lastly the string to be inserted is entered.
  5. **Search:**
     Finding out if an entered string is a substring of the main array (could be found with the same character order stuck together). Firstly the main string is entered then the other string to be searched for is entered.