Samir Varma

3/7/2025

Programming Methodology 1

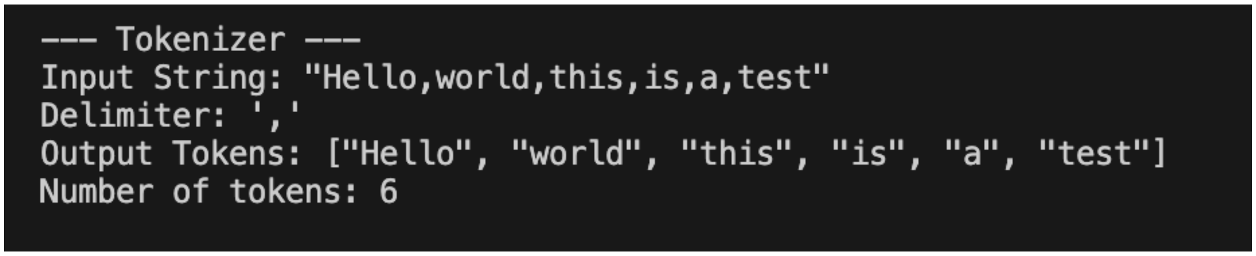
Professor Jorge Ortiz

Lab 5

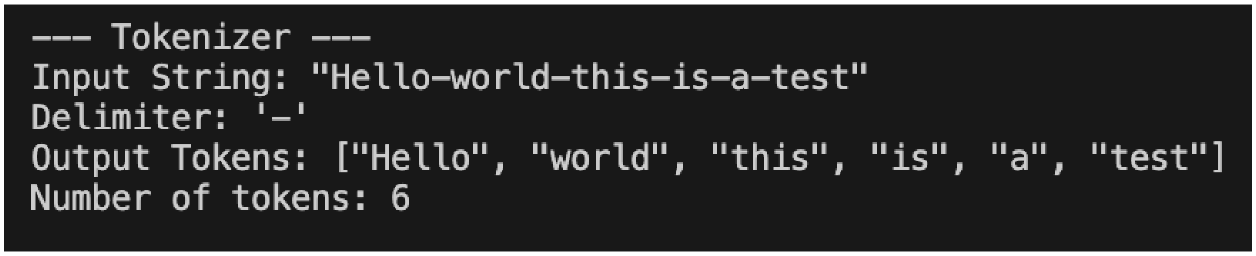
Exercise 1

Basic Operations

a) Using different delimiters

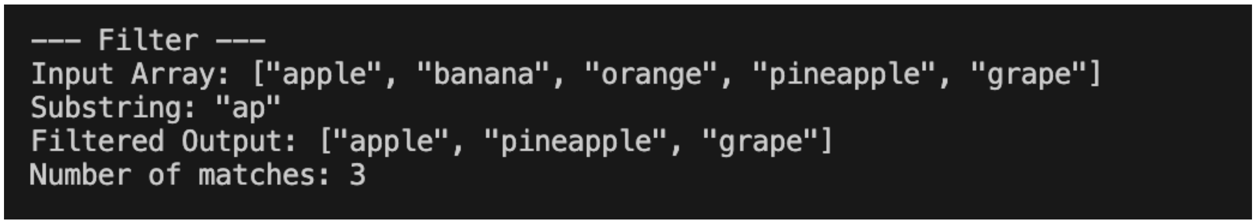


```
--- Tokenizer ---
Input String: "Hello,world,this,is,a,test"
Delimiter: ','
Output Tokens: ["Hello", "world", "this", "is", "a", "test"]
Number of tokens: 6
```

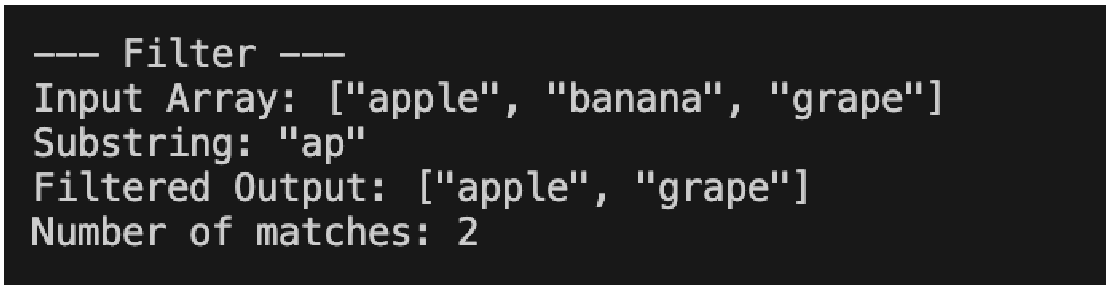


```
--- Tokenizer ---
Input String: "Hello-world-this-is-a-test"
Delimiter: '-'
Output Tokens: ["Hello", "world", "this", "is", "a", "test"]
Number of tokens: 6
```

b) Strings of different sizes

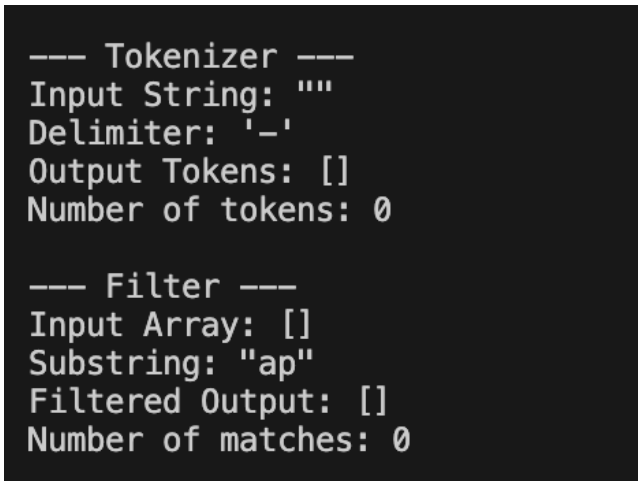


```
--- Filter ---
Input Array: ["apple", "banana", "orange", "pineapple", "grape"]
Substring: "ap"
Filtered Output: ["apple", "pineapple", "grape"]
Number of matches: 3
```



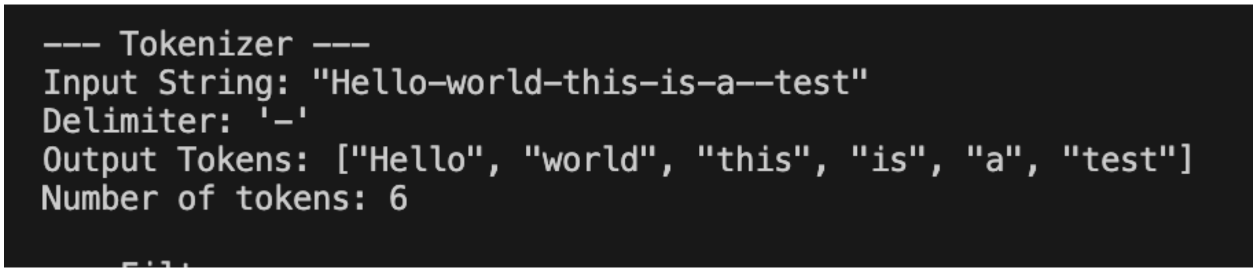
```
--- Filter ---
Input Array: ["apple", "banana", "grape"]
Substring: "ap"
Filtered Output: ["apple", "grape"]
Number of matches: 2
```

c) Empty strings and arrays



```
--- Tokenizer ---
Input String: ""
Delimiter: '-'
Output Tokens: []
Number of tokens: 0

--- Filter ---
Input Array: []
Substring: "ap"
Filtered Output: []
Number of matches: 0
```

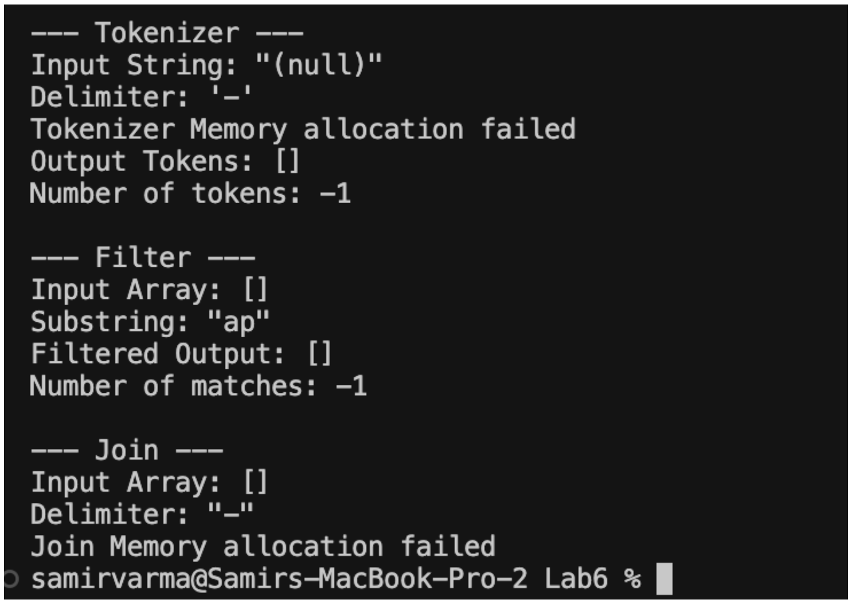
d) Repeated delimiter



```
--- Tokenizer ---
Input String: "Hello-world-this-is-a--test"
Delimiter: '-'
Output Tokens: ["Hello", "world", "this", "is", "a", "test"]
Number of tokens: 6
```

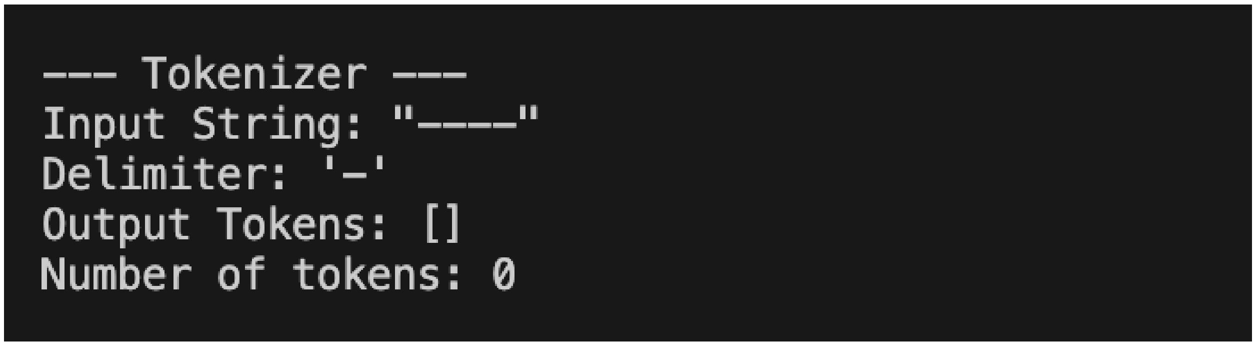
Edge Cases

a) Null inputs



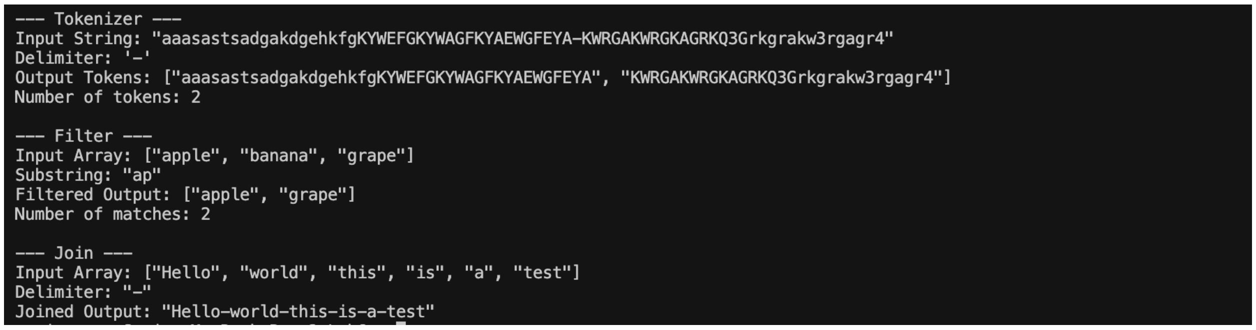
```
--- Tokenizer ---
Input String: "(null)"
Delimiter: '-'
Tokenizer Memory allocation failed
Output Tokens: []
Number of tokens: -1

--- Filter ---
Input Array: []
Substring: "ap"
Filtered Output: []
Number of matches: -1

--- Join ---
Input Array: []
Delimiter: "-"
Join Memory allocation failed
samirvarma@Samirs-MacBook-Pro-2 Lab6 %
```

b) Only delimiters



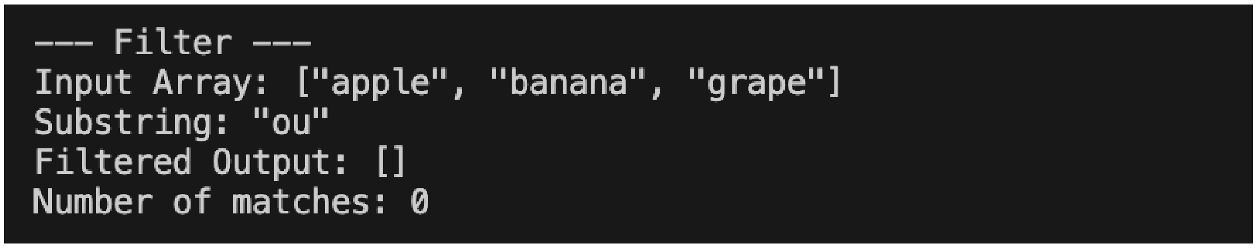
```
--- Tokenizer ---
Input String: "----"
Delimiter: '-'
Output Tokens: []
Number of tokens: 0
```

c) Very long strings



```
--- Tokenizer ---
Input String: "aaasastsadgakdgehkfgKYWEFGKYWAGFKYAEWGFEYA-KWRGAKWRGKAGRKQ3Grkgrakw3rgagr4"
Delimiter: '-'
Output Tokens: ["aaasastsadgakdgehkfgKYWEFGKYWAGFKYAEWGFEYA", "KWRGAKWRGKAGRKQ3Grkgrakw3rgagr4"]
Number of tokens: 2

--- Filter ---
Input Array: ["apple", "banana", "grape"]
Substring: "ap"
Filtered Output: ["apple", "grape"]
Number of matches: 2

--- Join ---
Input Array: ["Hello", "world", "this", "is", "a", "test"]
Delimiter: "-"
Joined Output: "Hello-world-this-is-a-test"
```

d) No matching substrings

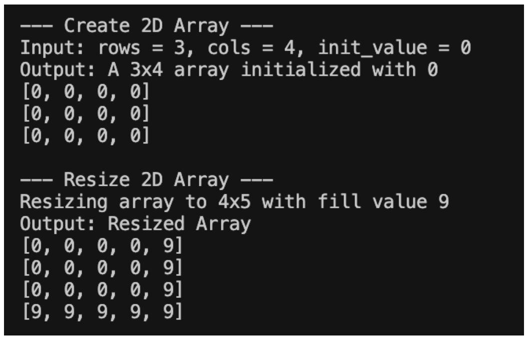


```
--- Filter ---
Input Array: ["apple", "banana", "grape"]
Substring: "ou"
Filtered Output: []
Number of matches: 0
```

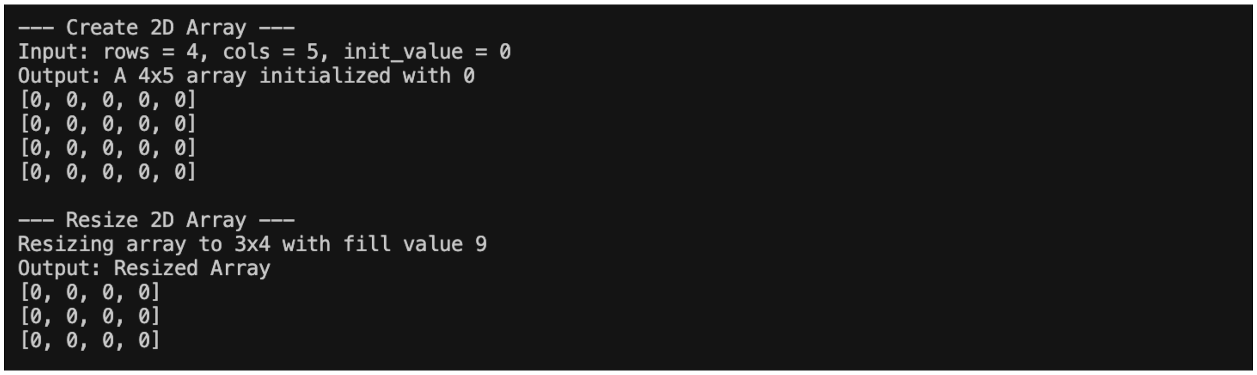
Exercise 2

Basic Operations

a) Create arrays of different sizes/resize to larger and smaller



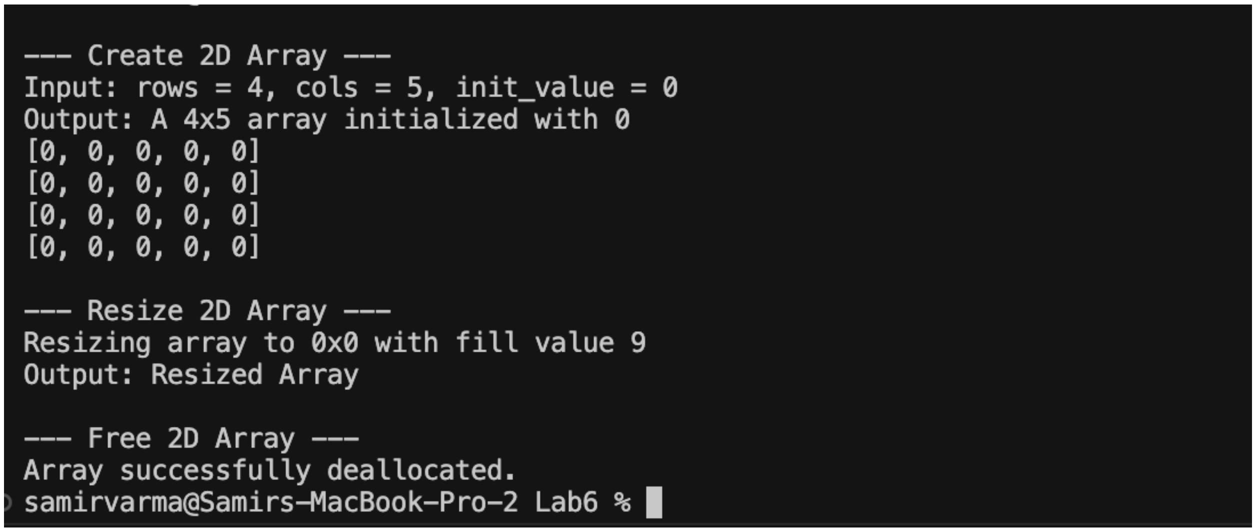
```
--- Create 2D Array ---
Input: rows = 3, cols = 4, init_value = 0
Output: A 3x4 array initialized with 0
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]

--- Resize 2D Array ---
Resizing array to 4x5 with fill value 9
Output: Resized Array
[0, 0, 0, 0, 9]
[0, 0, 0, 0, 9]
[0, 0, 0, 0, 9]
[9, 9, 9, 9, 9]
```



```
--- Create 2D Array ---
Input: rows = 4, cols = 5, init_value = 0
Output: A 4x5 array initialized with 0
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]

--- Resize 2D Array ---
Resizing array to 3x4 with fill value 9
Output: Resized Array
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]
```

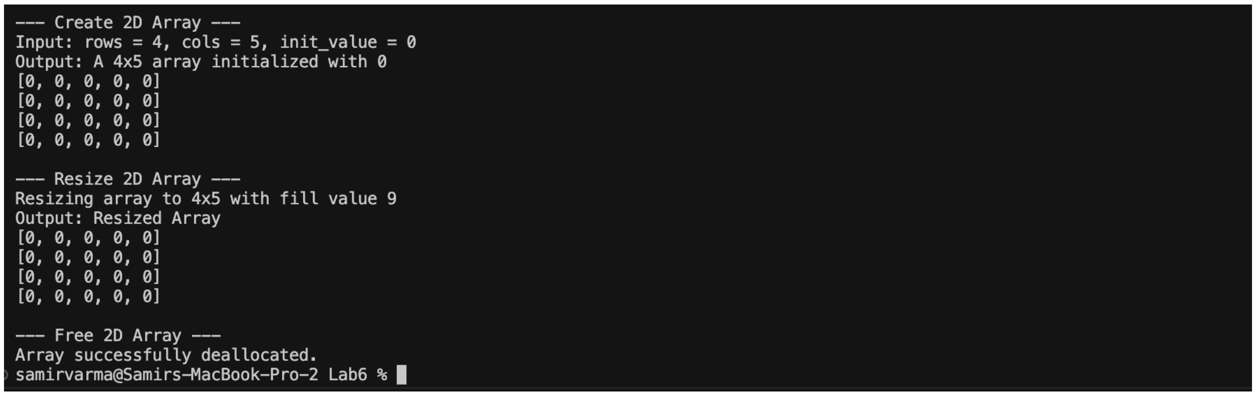
Edge Cases

a) Zero Dimensions/Resizing to 0 dimensions



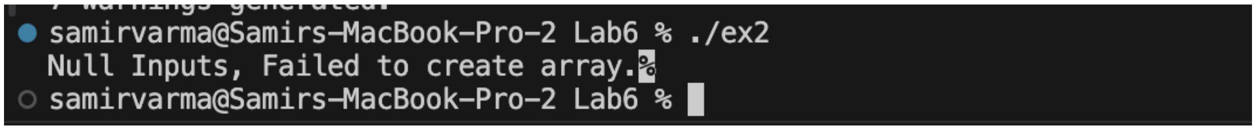
```
--- Create 2D Array ---
Input: rows = 4, cols = 5, init_value = 0
Output: A 4x5 array initialized with 0
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]

--- Resize 2D Array ---
Resizing array to 0x0 with fill value 9
Output: Resized Array

--- Free 2D Array ---
Array successfully deallocated.
samirvarma@Samirs-MacBook-Pro-2 Lab6 %
```

b) Resizing to same dimensions



```
--- Create 2D Array ---
Input: rows = 4, cols = 5, init_value = 0
Output: A 4x5 array initialized with 0
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]

--- Resize 2D Array ---
Resizing array to 4x5 with fill value 9
Output: Resized Array
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]

--- Free 2D Array ---
Array successfully deallocated.
samirvarma@Samirs-MacBook-Pro-2 Lab6 %
```

c) NULL Input

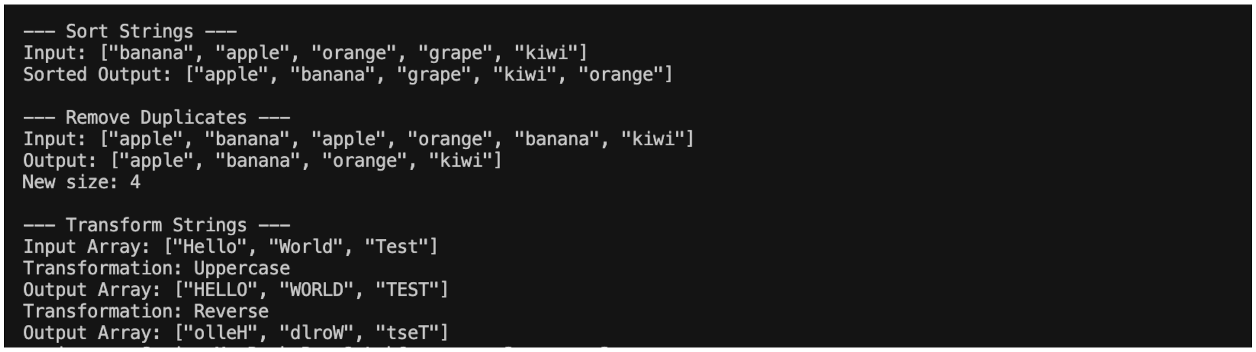


```
samirvarma@Samirs-MacBook-Pro-2 Lab6 % ./ex2
Null Inputs, Failed to create array.%
samirvarma@Samirs-MacBook-Pro-2 Lab6 %
```

Exercise 3

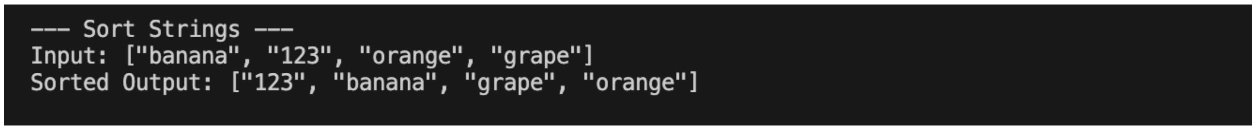
Basic Operations

a) Arrays with different length strings



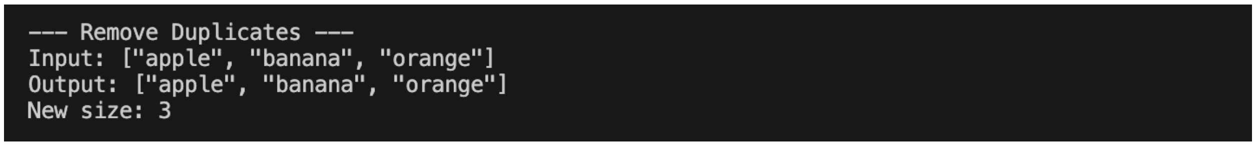
```
--- Sort Strings ---
Input: ["banana", "apple", "orange", "grape", "kiwi"]
Sorted Output: ["apple", "banana", "grape", "kiwi", "orange"]

--- Remove Duplicates ---
Input: ["apple", "banana", "apple", "orange", "banana", "kiwi"]
Output: ["apple", "banana", "orange", "kiwi"]
New size: 4

--- Transform Strings ---
Input Array: ["Hello", "World", "Test"]
Transformation: Uppercase
Output Array: ["HELLO", "WORLD", "TEST"]
Transformation: Reverse
Output Array: ["olleH", "dlroW", "tseT"]
```

b) Varying string content

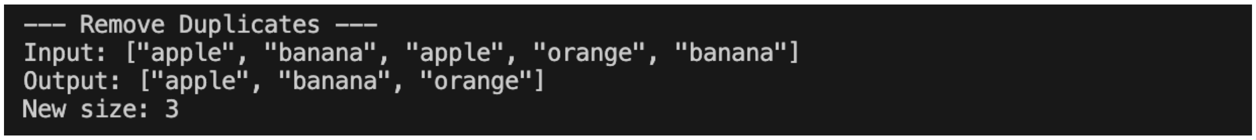


```
--- Sort Strings ---
Input: ["banana", "123", "orange", "grape"]
Sorted Output: ["123", "banana", "grape", "orange"]
```

c) With/Without duplicates

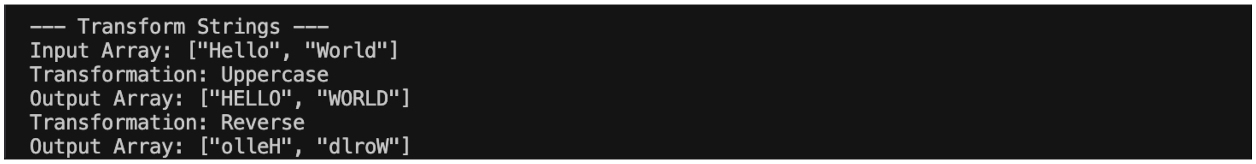


```
--- Remove Duplicates ---
Input: ["apple", "banana", "orange"]
Output: ["apple", "banana", "orange"]
New size: 3
```



```
--- Remove Duplicates ---
Input: ["apple", "banana", "apple", "orange", "banana"]
Output: ["apple", "banana", "orange"]
New size: 3
```

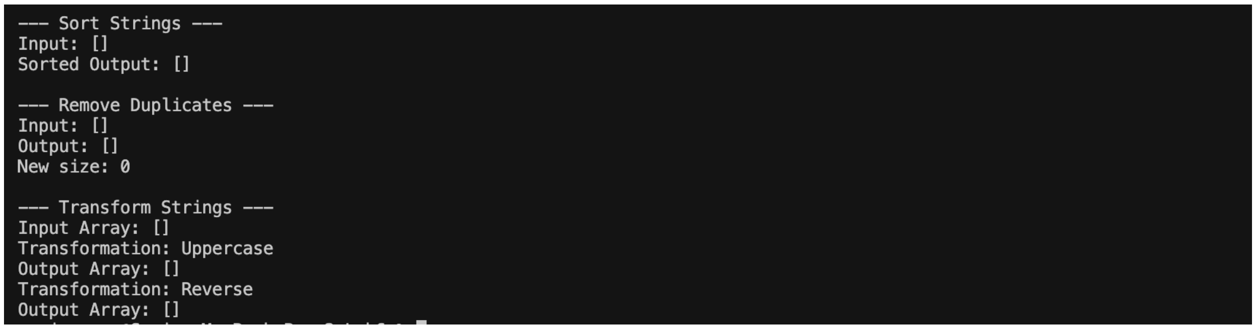
d) All transformations



```
--- Transform Strings ---
Input Array: ["Hello", "World"]
Transformation: Uppercase
Output Array: ["HELLO", "WORLD"]
Transformation: Reverse
Output Array: ["olleH", "dlroW"]
```

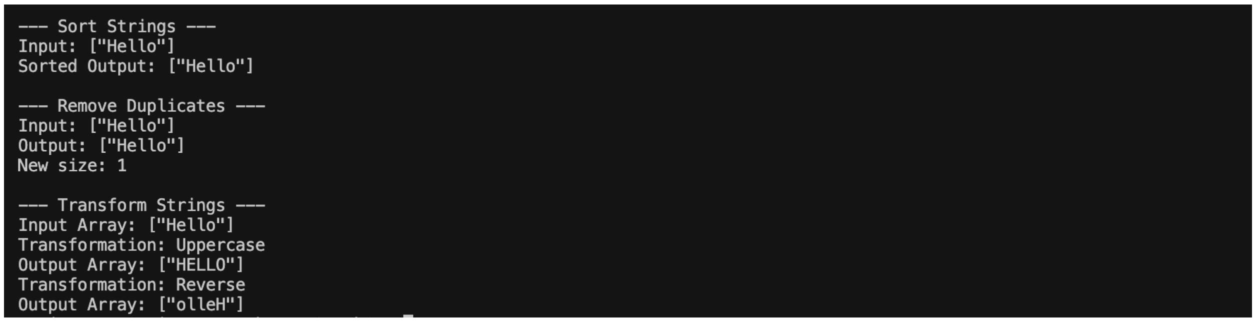
Edge Cases

a) Empty Strings



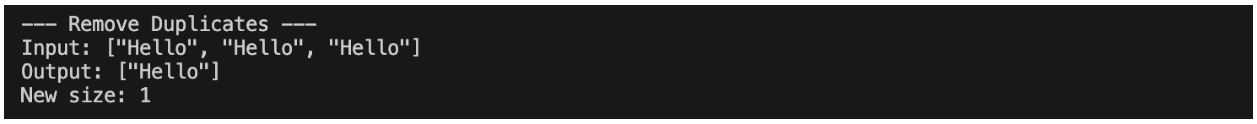
```
--- Sort Strings ---
Input: []
Sorted Output: []

--- Remove Duplicates ---
Input: []
Output: []
New size: 0

--- Transform Strings ---
Input Array: []
Transformation: Uppercase
Output Array: []
Transformation: Reverse
Output Array: []
```

b) Single element array



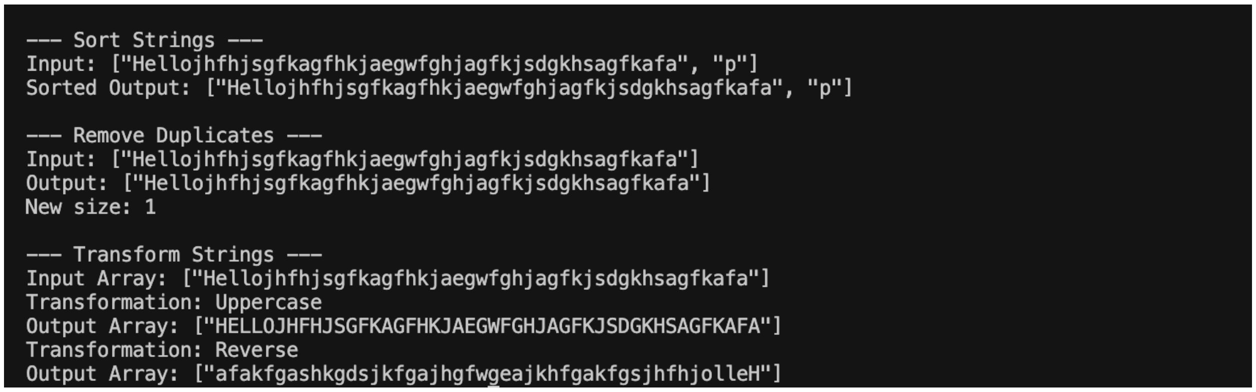
```
--- Sort Strings ---
Input: ["Hello"]
Sorted Output: ["Hello"]

--- Remove Duplicates ---
Input: ["Hello"]
Output: ["Hello"]
New size: 1

--- Transform Strings ---
Input Array: ["Hello"]
Transformation: Uppercase
Output Array: ["HELLO"]
Transformation: Reverse
Output Array: ["olleH"]
```

c) All duplicates



```
--- Remove Duplicates ---
Input: ["Hello", "Hello", "Hello"]
Output: ["Hello"]
New size: 1
```

d) Very long strings



```
--- Sort Strings ---
Input: ["Hellojhfhjsgfkagfhkjaegwfghjagfkjsdgkhsagfkafa", "p"]
Sorted Output: ["Hellojhfhjsgfkagfhkjaegwfghjagfkjsdgkhsagfkafa", "p"]

--- Remove Duplicates ---
Input: ["Hellojhfhjsgfkagfhkjaegwfghjagfkjsdgkhsagfkafa"]
Output: ["Hellojhfhjsgfkagfhkjaegwfghjagfkjsdgkhsagfkafa"]
New size: 1

--- Transform Strings ---
Input Array: ["Hellojhfhjsgfkagfhkjaegwfghjagfkjsdgkhsagfkafa"]
Transformation: Uppercase
Output Array: ["HELLOJHFHJSGFKAGFHKJAEGWFGHJAGFKJSDGKHSAGFKAFA"]
Transformation: Reverse
Output Array: ["afakfgashkgdsjkfgajhgfwgeajkhfgakfgsjhfhjolleH"]
```