1. **Problem Statement**

Make a string class that lets the user manipulate the string of data and execute many methods provided in the program. The program will display all the options that the user can select once he or she provides a string, which will be converted into and used as an array of characters.

1. **Requirements**
   1. **Assumptions**
      1. The user types in the appropriate number respective to the options provided in the command-line menu
      2. The user types the index number of the array when calling a function with position parameters. Ex: 1st character = position 0.
      3. User known that ONE space with no extra space around it is considered a character
      4. User enters either “1” or “2” for the instance number
   2. **Specifications**
      1. Welcome Message
      2. The program converts the string provided by the user into an array of characters
         1. Maximum of 25 characters
      3. Class called myString takes in the array of characters and modifies it according to user’s choice of options
      4. Create methods that allow modifying the input string
         1. size(), addStart(myString), addEnd(myString), partString(startPos, length), replPartString(myString, startPos), replWholeString(myString), emptyString(), fullString(), compareString(myString), charAt(pos), initString(), setString(string), getString(), printString().
      5. Constructor initializes myString data to empty string.
      6. Program and user actions and results are outputted to an output file names *myStringLog.txt*
         1. Displays any error inside this log file
      7. Thank You Message
2. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| String with maximum of 25 characters | Converts string in an array of characters | Command-line menu with options to modify the array |
|  | Check if all inputs are correct |  |
|  |  | “Welcome” & “Thank you” messages |
| A number respective to an action chosen by the user | Process the specific action using the array | Return or print out the result of the action selected by user |

1. **Test Strategy**
   1. Valid Data
   2. Invalid Data
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid | 1 | User enters more than 25 characters |  |  |  |  |
| Invalid | 2 | Method *addStart()* adds a string to have more than 25 characters in total |  |  |  |  |
| Valid | 1 | String is empty when calling method to find size |  |  |  |  |
| Invalid | 3 | When returning part of the string, given starting position < 0 |  |  |  |  |
| Invalid | 4 | When returning part of the string, given starting position = size |  |  |  |  |
| Invalid | 5 | When returning part of the string, given starting position > size |  |  |  |  |
| Invalid | 6 | When replacing characters, the final string exceeds 25 characters |  |  |  |  |
| Invalid | 7 | When returning character at position, position > end of string |  |  |  |  |
| Invalid | 8 | When returning character at position, position < 0 |  |  |  |  |
| Valid | 2 | When resetting string to null, string is already null |  |  |  |  |
| Valid | 3 | When printing myString data value, the myString is empty |  |  |  |  |
| Invalid | 9 | On the command-line menu, user enters a number outside of the set of options |  |  |  |  |

1. **Initial Algorithm**
   1. In main function
      1. Welcome message.
      2. Create two instances of myString variables
         1. Make the user add a string in each variable.
         2. Let the user manipulate the string data in each instance by calling methods in myString class
      3. Create an output file names *myStringLog.txt*
         1. Record each action and results in this file
         2. Record each errors in the program in this file
   2. In myString class
      1. Create constructor that will initialize myString data to empty string
      2. In size() function
         1. Set a counter variable *count* to 0
         2. For loop that iterates through the array until it results in null and in each step, it adds 1 to counter variable.
         3. Return *count*
      3. In addStart(myString) function
         1. Check that myString and currentString elements don’t add up to more than 25 characters
         2. Make a separate array named *tempArray*
         3. Copy the elements in currentString to *tempArray*
         4. Add myString elements in currentString after deleting everything from currentString
         5. Add back the original elements from *tempArray* to currentString at the end.
      4. In addEnd(myString) function
         1. Check that myString and currentString elements don’t add up to more than 25 charcters
         2. Find the total number of characters in currentString and using that number, add the myString elements at the end of currentString elements
      5. In partString(startPos, length) function
         1. Check startPos, let the user know and loop until the user gives a # inside bounds if
            1. startPos <0
            2. startPos = size
            3. startPos > size
         2. StartPos becomes the starting index number of the array
         3. Length + start becomes the ending index number of the array
         4. Using these two indices, return the set of characters inclusively between that location in the array
      6. In replPartString(myString, startPos) function
         1. Check if the number of elements in currentString and myString don’t add up to more than 25 characters
         2. startPos = index number of array
         3. Using a loop, set the rest of the index numbers to myString characters, which will automatically replace those characters.
      7. In replWholeString(myString) function
         1. Check that the number of elements in myString is not more than 25 characters
            1. If number of elements in myString is less than 25, set the rest of indices to null.
         2. Using a for loop, starting from 0, set each index of currentString to the corresponding index character of myString
      8. In emptyString() function
         1. Check and return true if each index value in the array is null.
            1. Else return false
      9. In fullString() function
         1. Check and return true if each index value is filled with a character.
            1. Else return false
      10. In compareString(myString) function
          1. Set a Boolean variable *temp* for each comparison
          2. Using for loop, compare each index of both myString and currentString
             1. If any of the indices do not match, set *temp* to false and return *temp*
      11. In charAt(pos) function
          1. Check that *pos* is not beyond end of string
          2. Check that *pos* is not negative
          3. Using *pos* as the index #, return the character at that position
      12. In initString() function
          1. Using for loop, go through each index and set everything to null
      13. In setString(string) function
          1. Using for loop, go through every data value in input string, and assign it to respective position in the array myString.
      14. In getString() function
          1. Convert the array myString in a string using a for loop that adds next character together each time it loop and return the string
      15. In printString() function
          1. Convert the array myString in a string using a for loop that adds next character together each time it loop and print the string
2. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid | 1 | User enters more than 25 characters | “I am entering more than 25 characters” | Code loops until correct number is given |  |  |
| Invalid | 2 | Method *addStart()* adds a string to have more than 25 characters in total | currentString: “Hello, ”  myString: “I am entering more than 25 characters” | Code loops until correct number is given |  |  |
| Valid | 1 | String is empty when calling method to find size | All data in myString are null. myString.size() is called | Returns 0 # of characters in the string |  |  |
| Invalid | 3 | When returning part of the string, given starting position < 0 | Position = -20 | Error. Loops until value is given between 0 and size of string |  |  |
| Invalid | 4 | When returning part of the string, given starting position = size and length given > 1 | Size = 11 characters  Position = 11  Length = 5  String = “Hello World” | Error. Loops until length given is 1 |  |  |
| Valid | 2 | When returning part of the string, given starting position = size and length = 1 | Size = 11 characters  Position = 11  Length = 1  String = “Hello World” | “d” |  |  |
| Valid | 3 | When returning part of the string, given starting position < size or > 0, and given length does not go beyond 25th character of string | Size = 11 characters  Position 7  Length 5  (*15 + 5 not greater than 20*)  String = “Hello World” | “World” |  |  |
| Invalid | 5 | When returning part of the string, given starting position > size | Position = 30 | Error. Loops until value is given between 0 and size of string |  |  |
| Invalid | 6 | When replacing characters, the final string exceeds 25 characters | myString: “Hello World”  replace with: “I am typing a string with more than 25 characters” | Error. Loops until the user provides small enough string that will <= 25 characters in total |  |  |
| Invalid | 7 | When returning character at position, position > end of string | Position: 30 | Error. Loops until value is given between 0 and size of string |  |  |
| Invalid | 8 | When returning character at position, position < 0 | Position: -20 | Error. Loops until value is given between 0 and size of string |  |  |
| Valid | 4 | When resetting string to null, string is already null | String: All null  Call function initString() | Tells the user, string null to begin with |  |  |
| Valid | 5 | When printing myString data value, the myString is empty | String: All null  Call function printString() | Tells the user, myString is empty |  |  |
| Invalid | 9 | On the command-line menu, user enters a number outside of the set of options | Option # selected: 100 | Error. Tells the user to select a valid option |  |  |

1. **Code**

myString.cpp

#include "stdafx.h"

#include "myString.h"

//class myString

//{

//public:

//myString();

//~myString();

//string getString();

//void setString(string);

//int size(); //returns integer of how many characters are in the string

//void addStart(myString); //adds the string in the input parameter to the front of current string

//void addEnd(myString); //adds the string in the input parameter to the end of the current string

//myString partString(int startPos, int length); //returns myString from current myString starting at startPos for length given

//void replPartString(myString, int startPos); //replaces characters starting at startPos with input string

//void replWholeString(myString); //replaces what is currently in string with input parameter string

//bool emptyString(); //reuturns true if myString contains exactly 25 charactes

//bool fullString(); //returns true if myString contains exactly 25 charactesrs

//bool compareString(myString); //returns true if both current value of string and input parameter string are equal

//char charAt(int pos); //returns character at that position

//void initString(); //resets/initializes string to null string

//void printString(); //prints myString data value to the monitor

//

//private:

///\*char myStringArray[25];\*/

//};

myString::myString()

{

for (int i = 0; i < 25; i++)

{

myStringArray[i] = NULL;

}

}

myString::~myString()

{

;

}

string myString::getString()

{

string result = "";

for (int i = 0; i < 25; i++)

{

result += myStringArray[i];

}

return result;

}

void myString::setString(string currentString)

{

for (int i = 0; i < 25; i++)

{

myStringArray[i] = NULL;

}

for (int i = 0; i < currentString.length(); i++)

{

myStringArray[i] = currentString.at(i);

}

}

int myString::size() //returns integer of how many characters are in the string

{

int check = 1;

int count = 0;

for (int i = 0; i < 25 && check != 0; i++)

{

if (i == 24 && myStringArray[i] != NULL)

{

count++;

}

else if (myStringArray[i] != NULL)

{

count++;

}

else if (myStringArray[i + 1] != NULL)

{

count++;

}

else {

check = 0;

}

}

return count;

}

void myString::addStart(myString input) //adds the string in the input parameter to the front of current string

{

int inputSize = input.size();

char tempArray[25];

for (int i = 0; i < 25; i++)

{

tempArray[i] = myStringArray[i];

myStringArray[i] = NULL;

}

for (int i = 0; i < inputSize; i++)

{

myStringArray[i] = input.charAt(i);

}

for (int i = inputSize; i < 25; i++)

{

myStringArray[i] = tempArray[i - inputSize];

}

}

void myString::addEnd(myString input) //adds the string in the input parameter to the end of the current string

{

int size = 0;

for (int i = 0; myStringArray[i] != NULL; i++)

{

size++;

}

for (int i = size; i < 25; i++)

{

myStringArray[i] = input.charAt(i - size);

}

}

myString myString::partString(int startPos, int length) //returns myString from current myString starting at startPos for length given

{

myString result;

char tempArray[25];

int count = 0;

result.initString();

for (int i = startPos; i < length; i++)

{

result.myStringArray[count] = myStringArray[i];

count++;

}

return result;

}

void myString::replPartString(myString input, int startPos) //replaces characters starting at startPos with input string

{

int size = input.size();

for (int i = startPos; i < size; i++)

{

myStringArray[i] = input.charAt(i);

}

}

void myString::replWholeString(myString input) //replaces what is currently in string with input parameter string

{

for (int i = 0; i < 25; i++)

{

if (input.charAt(i) == NULL)

{

myStringArray[i] = NULL;

}

else {

myStringArray[i] = input.charAt(i);

}

}

}

bool myString::emptyString() //reuturns true if myString contains exactly 25 charactes

{

int full = 0;

for (int i = 0; i < 25; i++)

{

if (myStringArray[i] != NULL)

{

full++;

}

}

if (full > 0)

{

return false;

}

else {

return true;

}

}

bool myString::fullString() //returns true if myString contains exactly 25 charactesrs

{

int empty = 0;

for (int i = 0; i < 25; i++)

{

if (myStringArray[i] == NULL)

{

empty++;

}

}

if (empty == 25)

{

return true;

}

else {

return false;

}

}

bool myString::compareString(myString input) //returns true if both current value of string and input parameter string are equal

{

bool result = true;

for (int i = 0; i < 25; i++)

{

if (myStringArray[i] != input.charAt(i))

{

result = false;

break;

}

}

return result;

}

char myString::charAt(int pos) //returns character at that position

{

return myStringArray[pos];

}

void myString::initString() //resets/initializes string to null string

{

for (int i = 0; i < 25; i++)

{

myStringArray[i] = NULL;

}

}

void myString::printString() //prints myString data value to the monitor

{

for (int i = 0; i < 25; i++)

{

cout << myStringArray[i];

}

cout << endl;

}

myString.h

#pragma once

#pragma once

#pragma once

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class myString

{

public:

myString();

~myString();

string getString();

void setString(string);

int size(); //returns integer of how many characters are in the string

void addStart(myString); //adds the string in the input parameter to the front of current string

void addEnd(myString); //adds the string in the input parameter to the end of the current string

myString partString(int startPos, int length); //returns myString from current myString starting at startPos for length given

void replPartString(myString, int startPos); //replaces characters starting at startPos with input string

void replWholeString(myString); //replaces what is currently in string with input parameter string

bool emptyString(); //reuturns true if myString contains exactly 25 charactes

bool fullString(); //returns true if myString contains exactly 25 charactesrs

bool compareString(myString); //returns true if both current value of string and input parameter string are equal

char charAt(int pos); //returns character at that position

void initString(); //resets/initializes string to null string

void printString(); //prints myString data value to the monitor

private:

char myStringArray[25];

};

Program\_1\_myString\_Home.cpp

#include "stdafx.h"

#include <iostream>

#include <string>

#include "myString.h"

using namespace std;

int getInstanceNumber();

int main()

{

int startPos, length, pos;

int instanceNumber;

ofstream outFile;

myString userInput;

string tempString;

bool quit = false;

int selection;

cout << "Greetings, user. My name is Nobody. Welcome to an amazing program that\nlets you select options and manipulate a string\nhow ever you want!" << endl;

myString instance1;

myString instance2;

outFile.open("myStringLong.txt");

outFile << "Program action - Create myString instane1" << endl;

outFile << "Program action - Create myString instane2" << endl;

do {

cout << "..............Menu.............." << endl;

cout << "1. Size" << endl;

cout << "2. Add a string in the beginning of current string" << endl;

cout << "3. Add a string in the end of current string" << endl;

cout << "4. Get part of the string from current string" << endl;

cout << "5. Replace part of the current string" << endl;

cout << "6. Replace the whole current string" << endl;

cout << "7. Check whether or not the string is empty" << endl;

cout << "8. Check whether or not the string is 100% full" << endl;

cout << "9. Compare a string with current string" << endl;

cout << "10. Get the character at a specific spot" << endl;

cout << "11. Reset string" << endl;

cout << "12. Assign a string to current string data value" << endl;

cout << "13. Get current string" << endl;

cout << "14. Print the string on your screen" << endl;

cout << "0. Quit" << endl;

cout << "................................" << endl << endl;

cout << "Please select an option: " << endl;

cin >> selection;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (selection)

{

case 1:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1) {

cout << "Size of the string is: " << instance1.size() << endl;

outFile << "Program action - Get size of string of instance1" << endl;

}

else {

cout << "Size of the string is: " << instance2.size() << endl;

outFile << "Program action - Get size of string of instance2" << endl;

}

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 2:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1) {

do

{

getline(cin, tempString);

cout << tempString;

cout << "What would you like to add in front of the string: " << endl;

getline(cin,tempString);

cout << tempString;

if (tempString.size() + instance1.size() > 25)

{

cout << "Sorry. The strings add up to be more than 25 characters in length. Try Again" << endl;

outFile << "User Action -ERROR: Strings added to have total of more than 25 characters in instance1" << endl;

}

} while (tempString.size() + instance1.size() > 25);

userInput.setString(tempString);

instance1.addStart(userInput);

outFile << "User Action - added string '" << tempString << "' in the front of current string of instance1" << endl;

}

else {

cout << "What would you like to add in front of the string: " << endl;

getline(cin,tempString);

do

{

cout << "What would you like to add in front of the string: " << endl;

getline(cin,tempString);

if (tempString.size() + instance2.size() > 25)

{

cout << "Sorry. The strings add up to be more than 25 characters in length. Try Again" << endl;

outFile << "User Action - ERROR: Strings added to have total of more than 25 characters in instance2" << endl;

}

} while (tempString.size() + instance2.size() > 25);

userInput.setString(tempString);

instance1.addStart(userInput);

outFile << "User Action - added string '" << tempString << "' in the front of current string of instance2" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 3:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1) {

do

{

cout << "What would you like to add in the end of the string: " << endl;

getline(cin,tempString);

if (tempString.size() + instance1.size() > 25)

{

cout << "Sorry. The strings add up to be more than 25 characters in length. Try Again." << endl;

outFile << "User Action - ERROR: Strings are added to have total of more than 25 characters in instance1" << endl;

}

} while (tempString.size() + instance1.size() > 25);

userInput.setString(tempString);

instance1.addEnd(userInput);

outFile << "User Action - added string '" << tempString << "' in the end of current string of instance1" << endl;

}

else {

do

{

cout << "What would you like to add in the end of the string: " << endl;

getline(cin,tempString);

if (tempString.size() + instance2.size() > 25)

{

cout << "Sorry. The strings add up to be more than 25 characters in length. Try Again." << endl;

outFile << "User Action - ERROR: Strings are added to have total of more than 25 characters in instance2" << endl;

}

} while (tempString.size() + instance2.size() > 25);

userInput.setString(tempString);

instance2.addEnd(userInput);

outFile << "User Action - added string '" << tempString << "' in the end of current string of instance2" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 4:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

do {

cout << "Start Position:" << endl;

cin >> startPos;

cout << "Length:" << endl;

cin >> length;

if (startPos < 0) {

cout << "Starting position is less than 0. Try Again." << endl;

outFile << "User Action - ERROR: Starting position is less than 0" << endl;

}

else if (startPos == length)

{

cout << "Starting position equals the length. Try Again." << endl;

outFile << "User Action - ERROR: Starting position equals length" << endl;

}

else if (startPos + length > 25) {

cout << "Starting position is greater than length. Try Again." << endl;

outFile << "User Action - ERROR: Starting position greater than length" << endl;

}

} while (startPos >= 0 || startPos != length || startPos <= length);

instance1.partString(startPos, length);

outFile << "Program Action - Returned part of the myString of instance1" << endl;

}

else {

do {

cout << "Start Position:" << endl;

cin >> startPos;

cout << "Length:" << endl;

cin >> length;

if (startPos < 0) {

cout << "Starting position is less than 0. Try Again." << endl;

outFile << "User Action - ERROR: Starting position is less than 0" << endl;

}

else if (startPos == length)

{

cout << "Starting position equals the length. Try Again." << endl;

outFile << "User Action - ERROR: Starting position equals length" << endl;

}

else if (startPos + length > 25) {

cout << "Starting position is greater than length. Try Again." << endl;

outFile << "User Action - ERROR: Starting position greater than length" << endl;

}

} while (startPos >= 0 || startPos != length || startPos + length > 25);

instance2.partString(startPos, length);

outFile << "Program Action - Returned part of the myString of instance2" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 5:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

do

{

cout << "What would you like to replace?" << endl;

getline(cin,tempString);

userInput.setString(tempString);

cout << "Starting Position:" << endl;

cin >> startPos;

if (tempString.size() + startPos > 25 && startPos < 25)

{

cout << "String is too big if we start at position. Try Again." << startPos << endl;

outFile << "User Action - ERROR: String is too big where start position is " << startPos << endl;

}

else if (startPos > 25)

{

cout << "Starting position out of bounds. Try entering between 0 and 25. Try Again." << endl;

outFile << "User Action - ERROR: Starting position out of bounds" << endl;

}

} while (tempString.size() + startPos > 25 && startPos < 25);

instance1.replPartString(userInput, startPos);

outFile << "User Action - Replaced part of the string of instance1 with '" << tempString << "'" << endl;

}

else {

do

{

cout << "What would you like to replace?" << endl;

getline(cin,tempString);

userInput.setString(tempString);

cout << "Starting Position:" << endl;

cin >> startPos;

if (tempString.size() + startPos > 25 && startPos < 25)

{

cout << "String is too big if we start at position. Try Again." << startPos << endl;

outFile << "User Action - ERROR: String is too big when start position is " << startPos << endl;

}

else if (startPos > 25)

{

cout << "Starting position out of bounds. Try entering between 0 and 25. Try Again." << endl;

outFile << "User Action - ERROR: Starting position out of bounds" << endl;

}

} while (tempString.size() + startPos > 25 && startPos < 25);

instance2.replPartString(userInput, startPos);

outFile << "User Action - Replaced part of the string of instance2 with '" << tempString << "'" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 6:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1) {

do

{

cout << "What would you like to replace?" << endl;

getline(cin,tempString);

if (tempString.length() > 25)

{

cout << "Length of your string is more than 25. Try Again." << endl;

outFile << "User Action - ERROR: Length of string more than 25" << endl;

}

} while (tempString.length() > 25);

userInput.setString(tempString);

instance1.replWholeString(userInput);

outFile << "User Action - Replaced whole string of instance1 with '" << tempString << "'" << endl;

}

else {

cout << "What would you like to replace?" << endl;

do

{

cout << "What would you like to replace?" << endl;

getline(cin,tempString);

if (tempString.length() > 25)

{

cout << "Length of your string is more than 25. Try Again." << endl;

outFile << "User Action - ERROR: Length of string more than 25" << endl;

}

} while (tempString.length() > 25);

instance1.replWholeString(userInput);

outFile << "User Action - Replaced whole string of instance2 with '" << tempString << "'" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 7:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

if (instance1.emptyString() == true)

{

cout << "The whole string is empty" << endl;

outFile << "Program Action - Returned true because string of instane1 is empty" << endl;

}

else {

cout << "The whole string is not empty" << endl;

outFile << "Program Action - Returned false because string of instane1 is not empty" << endl;

}

}

else {

if (instance2.emptyString() == true)

{

cout << "The whole string is empty" << endl;

outFile << "Program Action - Returned true because string of instane2 is empty" << endl;

}

else {

cout << "The whole string is not empty" << endl;

}

outFile << "Program Action - Returned false because string of instane2 is not empty" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 8:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

if (instance1.fullString() == true)

{

cout << "The whole string is full" << endl;

outFile << "Program Action - Returned true because string of instance1 has exactly 25 characters" << endl;

}

else {

cout << "The whole string is not full" << endl;

outFile << "Program Action - Returned false because string of instance1 does not have exactly 25 characters" << endl;

}

}

else {

if (instance2.fullString() == true)

{

cout << "The whole string is full" << endl;

outFile << "Program Action - Returned true because string of instance1 has exactly 25 characters" << endl;

}

else {

cout << "The whole string is not full" << endl;

outFile << "Program Action - Returned false because string of instance2 does not have exactly 25 characters" << endl;

}

}

cout << "...Done..." << endl;

system("pause");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 9:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

if (instance1.compareString(userInput) == true)

{

cout << "Both string values are equal" << endl;

outFile << "Program Action - Returned true because input string and current string of instance1 is exactly same" << endl;

}

else {

cout << "Both string values are not equal" << endl;

outFile << "Program Action - Returned false because input string and current string of instance1 is not exactly same" << endl;

}

}

else {

if (instance1.compareString(userInput) == true)

{

cout << "Both string values are equal" << endl;

outFile << "Program Action - Returned true because input string and current string of instance2 is exactly same" << endl;

}

else {

cout << "Both string values are not equal" << endl;

outFile << "Program Action - Returned false because input string and current string of instance2 is not exactly same" << endl;

}

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 10:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

do

{

cout << "Position: " << endl;

cin >> pos;

if (pos >= instance1.size())

{

cout << "Position entered is out of bounds. Try Again with position between 0 and 25" << endl;

outFile << "User Action - ERROR: Position entered is out of bounds for instance1" << endl;

}

else if (pos < 0)

{

cout << "Position entered is negative. Try Again." << endl;

outFile << "User Action - ERROR: Position entered is negative for instance1" << endl;

}

} while (pos >= instance1.size() || pos < 0);

cout << "Charater at position " << pos << " is " << instance1.charAt(pos) << endl;

outFile << "Program Action - Returned character " << instance1.charAt(pos) << " at position " << pos << " of instance1"<< endl;

}

else {

do

{

cout << "Position: " << endl;

cin >> pos;

if (pos >= instance1.size())

{

cout << "Position entered is out of bounds. Try Again with position between 0 and 25" << endl;

outFile << "User Action - ERROR: Position entered is out of bounds for instance1" << endl;

}

else if (pos < 0)

{

cout << "Position entered is negative. Try Again." << endl;

outFile << "User Action - ERROR: Position entered is negative for instance1" << endl;

}

} while (pos >= instance1.size() || pos < 0);

cout << "Charater at position " << pos << " is " << instance2.charAt(pos) << endl;

outFile << "Program Action - Returned character " << instance2.charAt(pos) << " at position " << pos << " of instance2" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 11:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

instance1.initString();

outFile << "Program Action - Resets string of instance1 to NULL" << endl;

}

else {

instance1.initString();

outFile << "Program Action - Resets string of instance2 to NULL" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 12:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

cout << "What would you like to assign to the string" << endl;

getline(cin,tempString);

instance1.setString(tempString);

outFile << "User Action - Assigns string " << tempString << "to current string of instance1" << endl;

}

else {

cout << "What would you like to assign to the string" << endl;

getline(cin,tempString);

instance1.setString(tempString);

if (instance1.emptyString()) {

outFile << "Program Actio - instance2 not updated" << endl;

}

outFile << "User Action - Assigns string " << tempString << "to current string of instance2" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 13:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

cout << instance1.getString() << endl;

outFile << "Program Action - Returned string '" << instance1.getString() << "' from instance1" << endl;

}

else {

cout << instance2.getString() << endl;

outFile << "Program Action - Returned string '" << instance2.getString() << "' from instance2" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 14:

instanceNumber = getInstanceNumber();

if (instanceNumber == 1)

{

instance1.printString();

outFile << "Program Action - Prints string of instance1 to the screen" << endl;

}

else {

instance2.printString();

outFile << "Program Action - Prints string of instance2 to the screen" << endl;

}

cout << "...Done..." << endl;

system("pause");

system("cls");

break;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

case 0:

quit = true;

system("cls");

break;

default:

cout << "Wrong input:" << endl;

break;

}

} while (quit == false);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cout << "Thank you for trying this program. See you again in next program" << endl;

outFile.close();

system("pause");

return 0;

}

int getInstanceNumber() {

int instanceNumber;

cout << "Which instance number would you like to work with? Type '1' or '2': " << endl;

cin >> instanceNumber;

return instanceNumber;

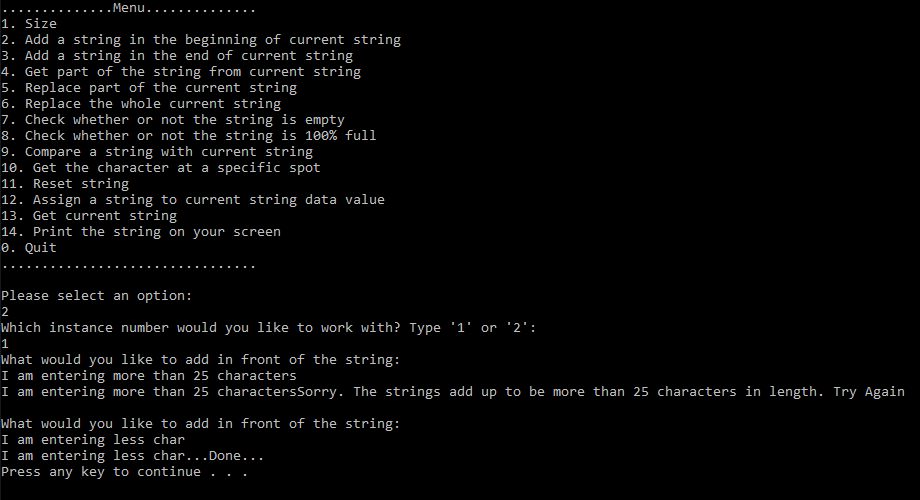
}

1. **Updated Algorithm**
   1. In main function
      1. Welcome message.
      2. Show a menu of options to the user
      3. Create two instances of myString variables
         1. Make the user add a string in each variable.
         2. Let the user manipulate the string data in each instance by calling methods in myString class
      4. Create an output file names *myStringLog.txt*
         1. Record each action and results in this file
         2. Record each errors in the program in this file
      5. Thank You message
   2. In myString class
      1. Create constructor that will initialize myString data to empty string
      2. In size() function
         1. Set a counter variable *count* to 0
         2. For loop that iterates through the array until it results in null and in each step, it adds 1 to counter variable.
         3. Return *count*
      3. In addStart(myString) function
         1. Check that myString and currentString elements don’t add up to more than 25 characters
         2. Make a separate array named *tempArray*
         3. Copy the elements in currentString to *tempArray*
         4. Add myString elements in currentString after deleting everything from currentString
         5. Add back the original elements from *tempArray* to currentString at the end.
      4. In addEnd(myString) function
         1. Check that myString and currentString elements don’t add up to more than 25 charcters
         2. Find the total number of characters in currentString and using that number, add the myString elements at the end of currentString elements
      5. In partString(startPos, length) function
         1. Check startPos, let the user know and loop until the user gives a # inside bounds if
            1. startPos <0
            2. startPos = size
            3. startPos > size
         2. StartPos becomes the starting index number of the array
         3. Length + start becomes the ending index number of the array
         4. Using these two indices, return the set of characters inclusively between that location in the array
      6. In replPartString(myString, startPos) function
         1. Check if the number of elements in currentString and myString don’t add up to more than 25 characters
         2. startPos = index number of array
         3. Using a loop, set the rest of the index numbers to myString characters, which will automatically replace those characters.
      7. In replWholeString(myString) function
         1. Check that the number of elements in myString is not more than 25 characters
            1. If number of elements in myString is less than 25, set the rest of indices to null.
         2. Using a for loop, starting from 0, set each index of currentString to the corresponding index character of myString
      8. In emptyString() function
         1. Check and return true if each index value in the array is null.
            1. Else return false
      9. In fullString() function
         1. Check and return true if each index value is filled with a character.
            1. Else return false
      10. In compareString(myString) function
          1. Set a Boolean variable *temp* for each comparison
          2. Using for loop, compare each index of both myString and currentString
             1. If any of the indices do not match, set *temp* to false and return *temp*
      11. In charAt(pos) function
          1. Check that *pos* is not beyond end of string
          2. Check that *pos* is not negative
          3. Using *pos* as the index #, return the character at that position
      12. In initString() function
          1. Using for loop, go through each index and set everything to null
      13. In setString(string) function
          1. Using for loop, go through every data value in input string, and assign it to respective position in the array myString.
      14. In getString() function
          1. Convert the array myString in a string using a for loop that adds next character together each time it loop and return the string
      15. In printString() function
          1. Convert the array myString in a string using a for loop that adds next character together each time it loop and print the string
      16. In getInstanceNumber()
          1. Ask the user which instance they want to work with
          2. Return the instance number as an int
2. **Test Plan Version 3**

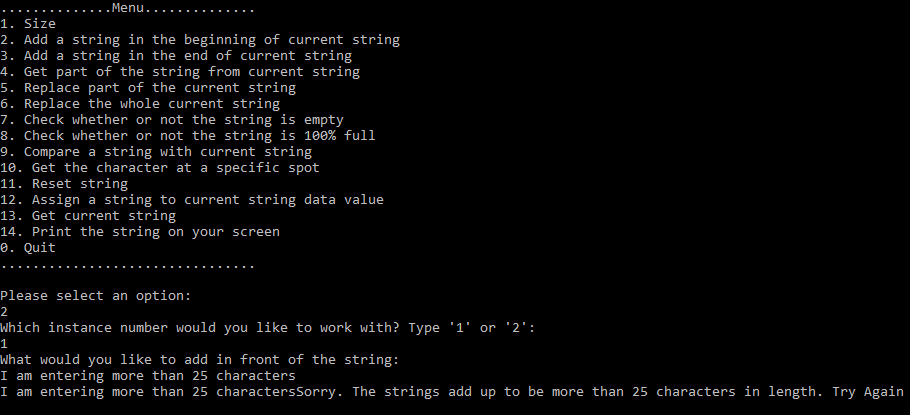
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid | 1 | User enters more than 25 characters | “I am entering more than 25 characters” | Code loops until correct number is given | Code loops until correct number is given | Pass |
| Invalid | 2 | Method *addStart()* adds a string to have more than 25 characters in total | currentString: “Hello, ”  myString: “I am entering more than 25 characters” | Code loops until correct number is given | Code loops until correct number is given | Pass |
| Valid | 1 | String is empty when calling method to find size | All data in myString are null. myString.size() is called | Returns 0 # of characters in the string | Returns 0 # of characters in the string | Pass |
| Invalid | 3 | When returning part of the string, given starting position < 0 | Position = -20  Length = 5 | Error. Loops until value is given between 0 and size of string | Error. Keeps looping | Fail |
| Invalid | 4 | When returning part of the string, given starting position = size and length given > 1 | Size = 11 characters  Position = 11  Length = 5  String = “Hello World” | Error. Loops until length given is 1 | Error. Keeps looping | Fail |
| Valid | 2 | When returning part of the string, given starting position = size and length = 1 | Size = 11 characters  Position = 11  Length = 1  String = “Hello World” | “d” | Error. Keeps looping | Fail |
| Valid | 3 | When returning part of the string, given starting position < size or > 0, and given length does not go beyond 25th character of string | Size = 11 characters  Position 7  Length 5  (*15 + 5 not greater than 20*)  String = “Hello World” | “World” | Error. Keeps Looping | Fail |
| Invalid | 5 | When returning part of the string, given starting position > size | Position = 30 | Error. Loops until value is given between 0 and size of string | Error. Keeps Looping | Fail |
| Invalid | 6 | When replacing characters, the final string exceeds 25 characters | myString: “Hello World”  replace with: “I am typing a string with more than 25 characters” | Error. Loops until the user provides small enough string that will <= 25 characters in total | Error. Does not let user type in anything | Fail |
| Invalid | 7 | When returning character at position, position > end of string | Position: 30 | Error. Loops until value is given between 0 and size of string | Error. Loops until value is given between 0 and size of string | Pass |
| Invalid | 8 | When returning character at position, position < 0 | Position: -20 | Error. Loops until value is given between 0 and size of string | Error. Loops until value is given between 0 and size of string | Pass |
| Valid | 4 | When resetting string to null, string is already null | String: All null  Call function initString() | Still resets to NULL | Still resets to NULL | Pass |
| Valid | 5 | When printing myString data value, the myString is empty | String: All null  Call function printString() | Nothing is printed on the screen | Nothing is printed on the screen | Pass |
| Invalid | 9 | On the command-line menu, user enters a number outside of the set of options | Option # selected: 100 | Error. Tells the user to select a valid option | Error. Tells the user to select a valid option | Pass |
| Invalid | 10 | For getting part string, starting position is less than 0 | startPos = -10 | Error. Tells the user to try again | Error. Tells the user to try again | Pass |
| Invalid | 11 | For getting part string, starting position equals length of string | startPos = 11  length = 11  string: “hello world” | Error. Tells the user to try again | Error. Tells the user to try again | Pass |
| Invalid | 12 | For getting part string, starting position is greater than length | startPos = 40  length = 11  string: “hello world” | Error. Tells the user to try again | Error. Tells the user to try again | Pass |
| Invalid | 13 | When replacing part string, input string is too big where the position is | startPos = 8  length = 11  currentString: “hello world”  input string: “ke” | Error. Tells the user to try again | Error. Does not let user to type in the string | Fail |
| Invalid | 14 | When replacing part string, start position is more than 25 | startPos = 40  length = ll  currentString: “hello world”  input string: “ke” | Error. Tells the user to try again | Error. Tells the user to try again | Pass |
| Invalid | 15 | When replacing whole string, length of input string is more than 25 | Length of input string: 26  String: “abcdefghijklmnopqrstuvwxyz” | Error. Tells the user to try again | Error. Does not let user to type in the string | Fail |

1. **Screenshots**

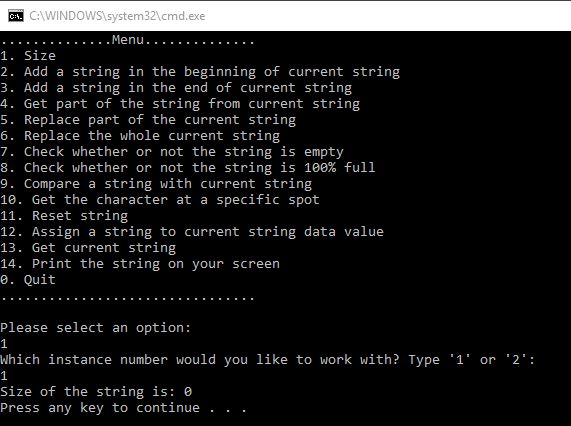
Invalid Test Case 1:



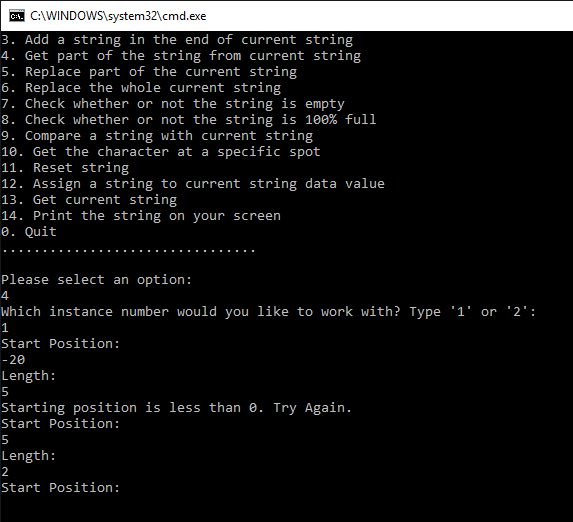
Invalid test Case 2:



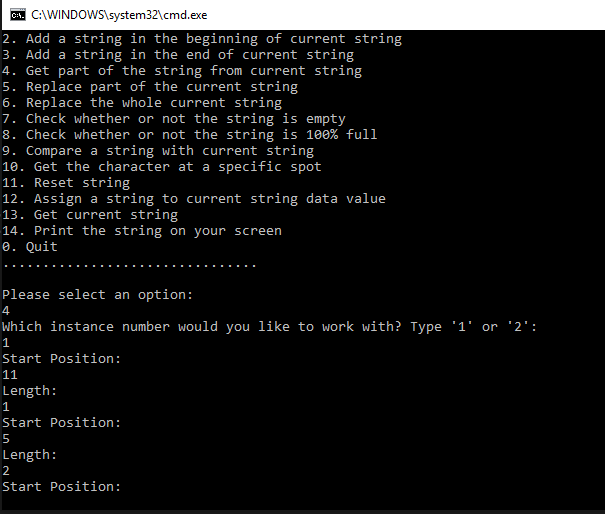
Valid Test Case 1:



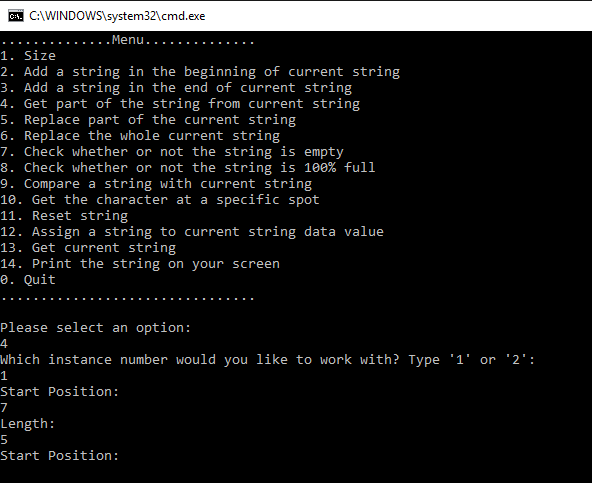
Invalid Test Case 3:



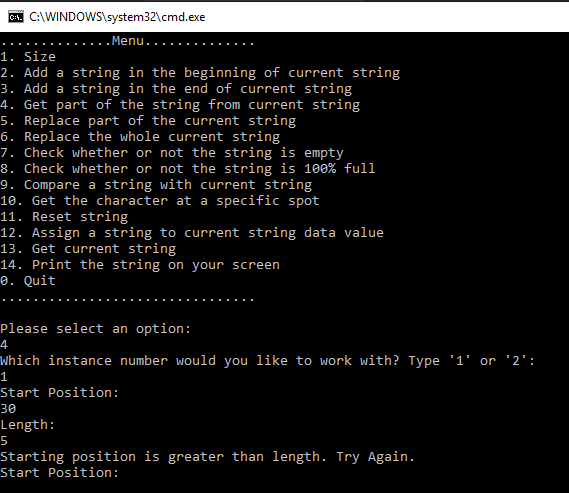
Valid Test Case 2:



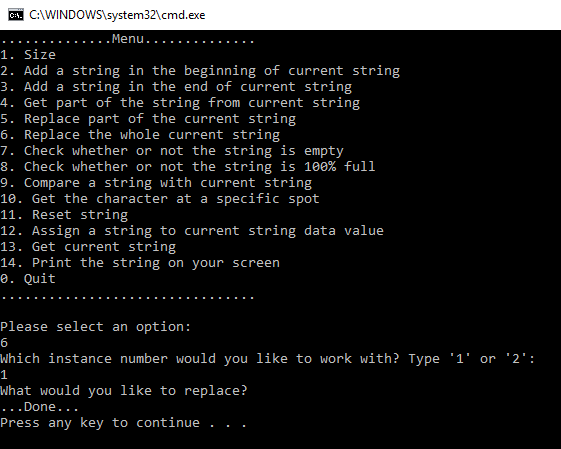
Valid Test Case 3:



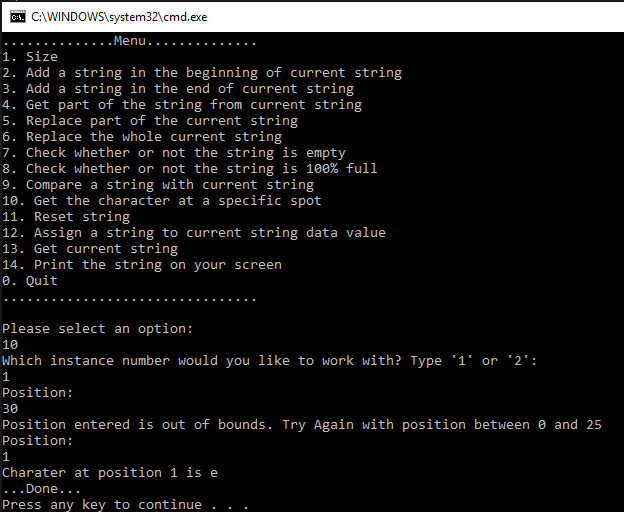
Invalid Test Case 5:



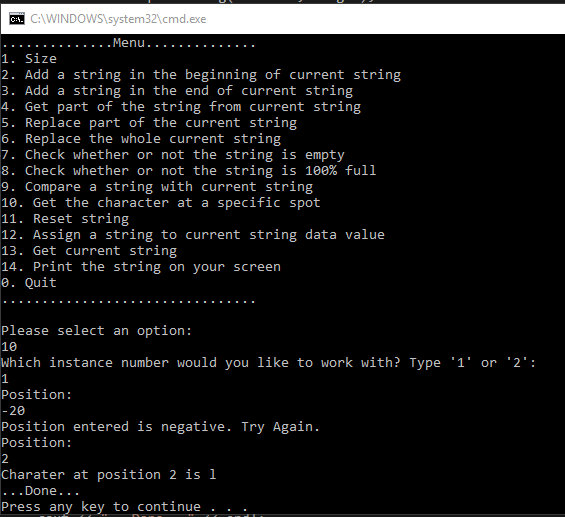
Invalid Test Case 6:



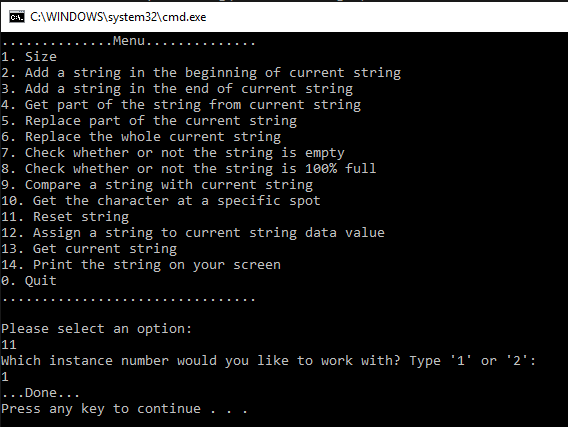
Invalid Test Case 7:

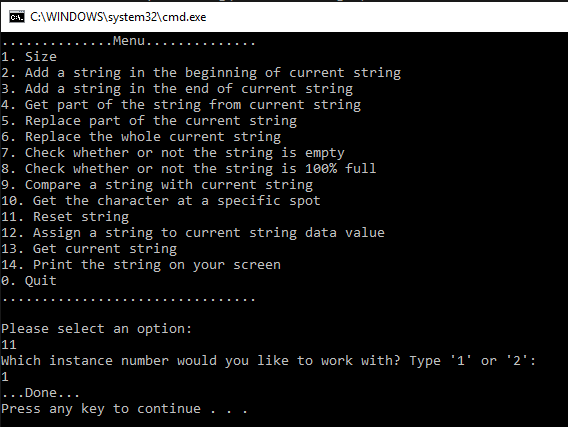


Invalid Test Case 8:

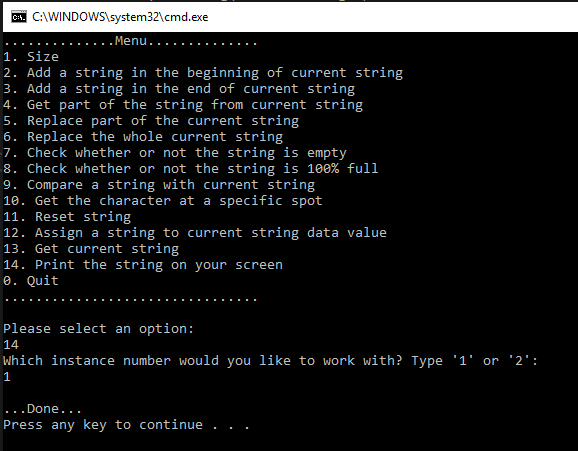


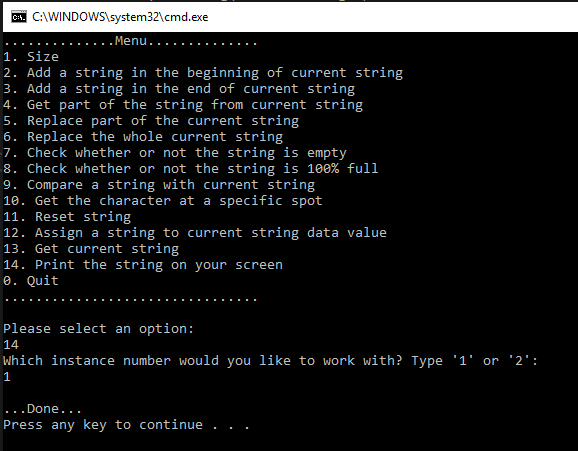
Valid Test Case 4:



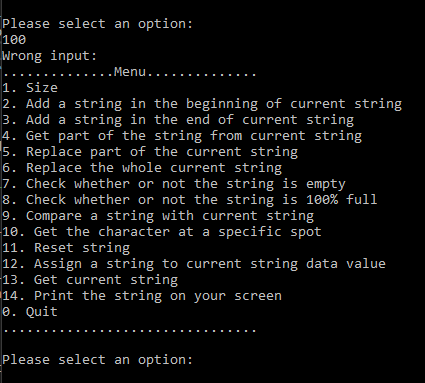


Valid Test Case 5:

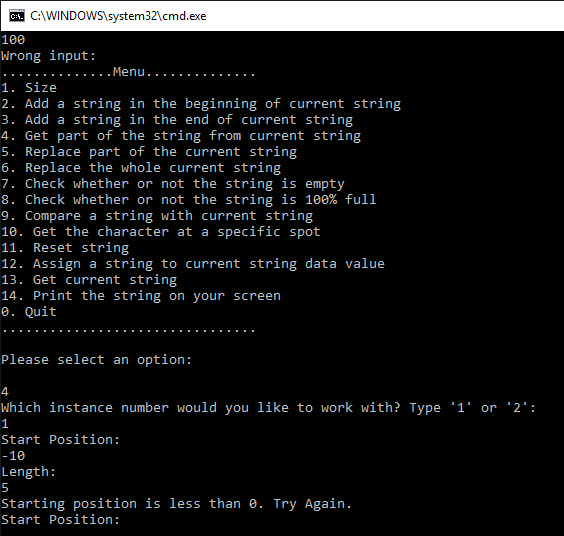




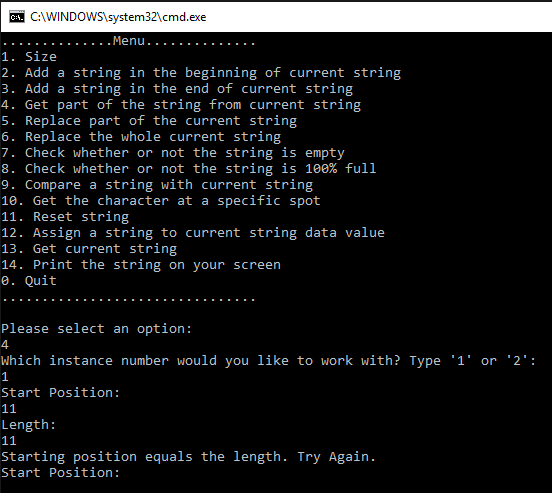
Invalid Test Case 9:



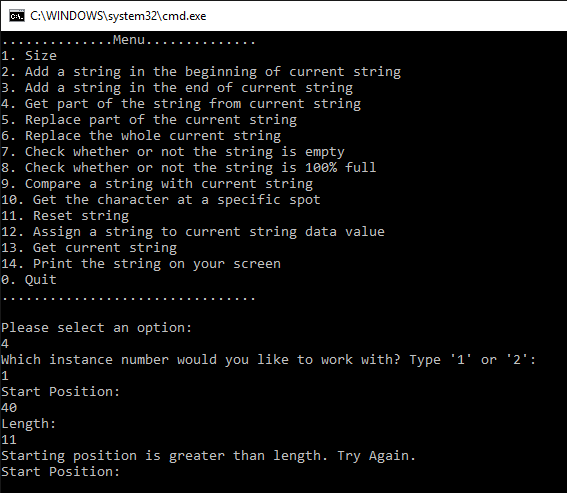
Invalid Test Case 10:



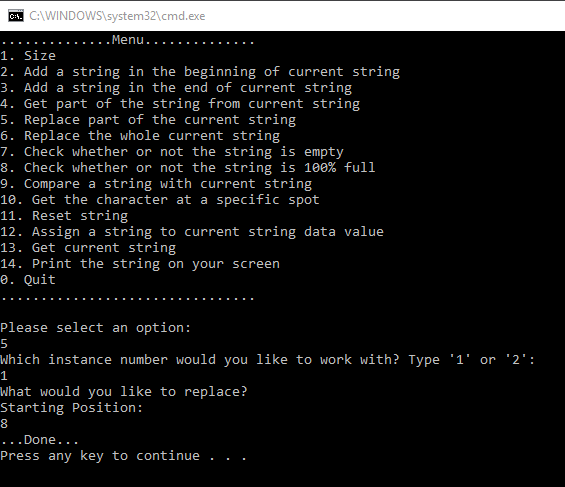
Invalid Test Case 11:



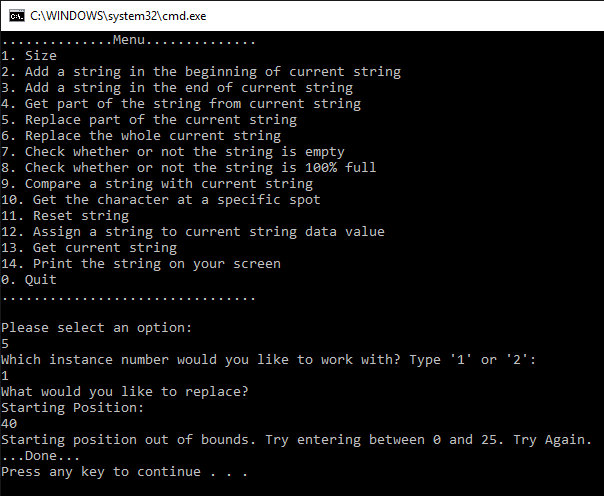
Invalid Test Case 12:



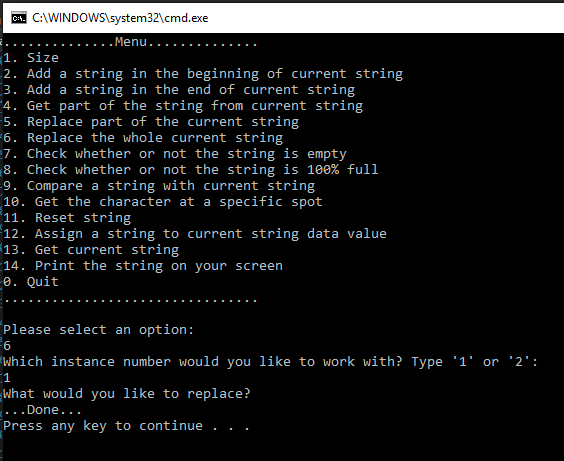
Invalid Test Case 13:



Invalid Test Case 14:



Invalid Test Case 15:



1. **Status**

Getting data works most of the time. Setting data to particular part of the string does not work all the time. The errors successfully are written in the file. Other than that, code works fine.