Nahrin Sharna

Professor Robert Mann

CIS 200

Project 01

Date:02/18/2019

1. **Problem Statement:** The program requires to create a string class myString to store a string. The string that should be stored can have size of maximum 25. All of the output should be displayed as well as to be stored in an output file.
2. **Requirements:**
3. Assumptions: Since string library can not be used mostly, an array of character should be used to store the string.

The size of the array is constant value of 25.

1. Specification:
2. myString(): Pre: Declare the variable to store string and initialize it

Post: Variable declared and initialized

1. size(): Pre: string has a size where: size>25

Post: size of the string is returned

1. addStart(myString) – Pre: Implicit and parameter string are of myString object where, size of each string is less than 25

Post: parameter string is added in front of implicit string when total sizes are less than 25; if total size > 25, show error message

1. addEnd(myString) – Pre: Implicit and parameter string are of myString object where, size of each string is less than 25

Post: parameter string is added at the end of implicit string when total sizes are less than 25; if total size > 25, show error message

1. partString(startPos, length) : Pre: anString is a string of characters with a length

Post: Characters from startPos are returned or Show appropriate message

1. replPartString(myString, startPos) – Pre: anString is a string of characters with a length

Post: characters starting at startPos are replaced with input string

1. replWholeString (myString) – myString s is a string of characters

Post: current string data value is replaced with input parameter string

1. compareString(myString) – Pre: Implicit and Parameter strings are strings of character array

Post: Values of both strings are compared and show appropriate message

1. initString() – Pre: anString is a string of character array[size], size <25

Post: string is initialized to null string

1. setString(string) – Pre: anString is a string of character array[size], size <25

Post: string of size < 25 is assigned to myString data value

1. getString() – Pre: anString is a string of character array[size], size <25 Post: string of data from myString data value is returned
2. printStringScreen() – Pre: anString is a string of character array[size], size <25

Post: prints myString data value to the monitor

1. numericString() – Pre: anString has a data value with a size, where size<25

Post: returns Boolean telling if data value is either integer or real (signs, decimal point. etc.)

1. alphabeticString() – Pre: anString has a data value with a size, where size<25
2. Post: returns Boolean telling if data value is all alphabetic characters
3. **Decomposition Diagram:**

Main

Output

Input

Process

int size();

myString()

printStringScreen()

void addStart()

getString()

setString(string s)

void addEnd()

char\* partString()

char\* replPartString()

void replWholeString()

bool compareString()

void initString()

Bool numericString()

Bool alphabeticString()

1. **Algorithms:**

**Header file(myString.h)**

//class myString definition

//private

//declaration of array of character – char a[];

//public:

//default constructor myString()declaration

// void set method declaration with a string parameter

//string type get method declaration with no parameter

// int type size() function declaration

//void addStart(myString) function declaration

// void addEnd(myString) function declaration

//void partString(startPos, length) function declaration

//void replPartString(myString, startPos) function declaration

//void replWholeString( myString) function declaration

// boolean type compareString(myString) function declaration

// void initString() function declaration

// void printStringScreen() function declaration

// boolean type of numericString() function declaration

// boolean type of alphabeticString() function declaration

**Implementation File:**

//include header file

//constant variable size of Max declared with the value of 25

// myString() {

//initialize the array of character to it’s null position

}//end of function

// void setString(string s) {

//for loop x = 0 to less than length of s

// store each character of s to char a[]

} // end of function

// string getString() {

// return s;

} //end of function

// int size() {

//declare and initialize variable size equal to zero

// for loop i = 0 to less then Max

{

// if character array a[i] is not null, increment size by 1

}

//return size;

}//end of function

//void addStart(myString st) {

//this function adds the string in the input parameter to the front of current string

//check if the size of st + size of current string is more than 25

// show an error message

//otherwise, when total size is less than 25

// create a new array of char with the new total size

// for loop x = 0 to size of the char

// assign each character of string s to the new array of char in the position of x

// for loop x = 0 to size of the current string

// assign each character of current string

}// end of the function

// void addEnd(myString st) {

// this function adds the string in the input parameter to the end of the current string

//check if the size of st + size of current string is more than 25

// show an error message

// otherwise, when total size is less than 25

// create a new array of char with the new total size

// for loop x = 0 to size of the char

// assign each character of current string to the new array of char in the position of x

// for loop x = 0 to size of new parameter string

// assign each character of parameter string after the current string characters

} // end of the function

//void partString(int startPos, int length) {

//this function returns string from startPos for length given

//(handle startPos < 0, startPos = size returns null string, handle startPos > size)

// if startPos is negative, show an error message

// else if startPos is equal to size of a string {

// if string is empty, shows a message

// otherwise, shows another message saying starting point and size is equal

}

// else if startPos is greater than the length, show an error message

// otherwise, for loop from x = startPos to length

// print each character of the string from the position of x

} // end of function

//void replPartString(myString st, int startPos) {

//this function replaces characters starting at startPos with input string

//copy and assign input string to the starting position of the current string

}// end of function

//void replWholeString( myString st)

{

// replaces current string data value with input parameter string

//copy and assign the parameter string to the current string

}

// boolean type compareString(myString st) {

//– compare current value of string with input parameter string

// if the size of input string and current is not equal

//return false

// otherwise when the sizes are equal:

//if each character of input and current string is same

//return true

// otherwise, return false

}//end of function

// void initString() {

// this function resets / initializes string to null string

// assign current string(array of character) is equal to its NULL position

}//end of function

// void printStringScreen() {  
 //this function prints myString data value to the monitor

//for int i = 0 to the size of myString array size

// print each character of array

}//end of function

//bool numericString() {

// returns Boolean telling if data value is either integer or real(signs, decimal point.etc.)

// Checks if current index position of the string is not a digit

//return false

//otherwise return true

}

//bool alphabeticString() {

//– returns Boolean telling if data value is all alphabetic characters

// for i = 0 to the size of the array {

// if element of array at the position i is equal to isalpha, return true

// otherwise return false

}

}

**Main file:**

//include iostream

//include header file

//include fstream library

//using namespace std

// main function () {

// declare 3 object variables – “first”, “second,” “third” of myString class

//set values for each of 3 objects using set method

// print the values of each objects using get method

// call the function size() to find the size of a variable and print it

// add the “first” string infront of the second string to get a new “second” myString object using second.addStart(first) function call

// add the “third” string at the end of the first string and get a new value for “first” using first.addEnd(third) function call

//prompt to return the first string from a negative position by calling partString() function

//prompt to return the first string from a position which is greater than the string size using partString() function

//prompt to return the fourth string from a position of fourth string size when fourth string is NULL by calling partString() function

// prompt to return the first string from a position which is equal to the size of the first string by using partString() function and the size is not zero

// replPartString(myString, startPos) call this function

// replace the whole first string object with the third one and get a new first string value by calling replWholeString( myString) with passing the third myString object as a parameter

//compare the value of current first string object = “third” with third string object = “third” by calling Boolean compareString(myString) function while passing third myString object as a parameter

//this should return true and thus print a message

// compare the value of current first string object = “third” with second string object = “first second” by calling Boolean compareString(myString) function while passing second myString object as a parameter

//this should return false and thus print an appropriate message

//reset the value of first string object = “third” to its NULL value by calling initString() function for first string object

//print the current third string object value by calling printStringScreen() function for third string object

//To check if the third string object = “third” has any digit or real, call Boolean numericString()

//this should return false and display an appropriate message.

//set the third string object with a value = “7234” by using setString method that will update the value of third string object

//To check if the current third string object = “third” has any digit or real, call Boolean numericString()

//this should return false and thus show another appropriate message in the display

//To check if the third myString object = “7234” has all the alphabetic value, call boolean alphabeticString() function

// this should return false and thus show an appropriate message

//To check if the second myString object = “first second” has all the alphabetic value, call boolean alphabeticString() function

// this should return true and thus show another appropriate message

**//** system pause

//return 0

//end of main function

1. **Initial Test Plan:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | Use set method to insert value in one of the object of class myString  Set a string that has a size less than 25  To check, use get method to check the value for that object and print it. | first.setString(“First”);  first.getString(); | First string is: First |  |  |
| 2 | Valid | Use size function to get the size of a string and print the size | myString first = “first”  first.size(); | The size of the first string is: 5 |  |  |
| 3 | Valid | Add first string in front of second string  The sum of the sizes of both strings is less than 25 | myString first = “First”  myString second = “Second” | After operation, the new second string is = First Second |  |  |
| 4 | Valid | Add third string at the end of first string  The sum of the sizes of both strings is less than 25 | myString first = “First”  myString third = “Third” | After operation, the new first string is = First Third |  |  |
| 5 | Invalid | Return the first string value from a negative starting position | myString first = “First Third”  position = -1 | Error: Starting point cannot be negative. |  |  |
| 6 | Invalid | Return the first string value from a starting position which is more than string length | myString first = “First Third”  position = 15 | ERROR: Starting position exceeding string length |  |  |
| 7 | Valid | Returning a string from a starting position which is equal to the size of the string  The string is NULL | myString fourth = “”  position = fourth.size() | Empty String |  |  |
| 8 | Valid | Returning a string from a starting position which is equal to the size of the string  The string is not NULL | myString first = “First Third”  Position = 11 | Starting point is equal to size.  After returning, the first string is: “First Third” |  |  |
| 9 | Valid | Replace the character of second string from a valid position with the parameter string | second = “First Second”  startPos = 2  third = “Third” | After replacing, the new second string is: Thirdrst Second |  |  |
| 10 | Valid | Replace the whole first string with third string and get a new first string | myString first = “First Third” | After replacing, the new first string is: “Third” |  |  |
| 11 | Valid | Compare the value of current myString first and myString third | myString first = “Third”  myString third = “Third” | Both are equal |  |  |
| 12 | Valid | Compare the value of current myString first and myString second | myString first = “Third”  myString second= “First Second” | Not equal |  |  |
| 13 | Valid | Reset a string to its Null value | myString first = “Third” | After resetting to its null value, First string is: |  |  |
| 14 | Valid | Print the current value of a string using printStringScreen() | myString third = “Third” | Third String = Third |  |  |
| 15 | Valid | Use numericString() to check if the string has digits or real | myString third = “Third” | Data value has no integer |  |  |
| 16 | Valid | Use numericString() to check if the string has digits or real | myString second = “7234” | Data value has integer or real |  |  |
| 17 | Valid | Use alphabeticString() too check if a string has all alphabets. | myString third = “Third” | Data value contains all alphabets. |  |  |
| 18 | Valid | Use alphabeticString() too check if a string has all alphabets. | myString second = “7234” | Data value does not have all alphabets |  |  |

1. **UML:**

|  |
| --- |
| **myString** |
| -str: char \* |
| +myString()  +setString(s: string): void  +getString(): string  +size(): int  +addStart(st: myString): void  +addEnd(st: myString): void  +partString(startPos: int, len: int): char\*  +replPartString(s: myString, startPos: int): char\*  +replWholeString(st: myString): void  +compareString(st: myString): bool  +initString(): int  +printStringScreen(): void  +numericString(): bool  +alphabeticString(): bool |

1. **Source Code:**

**Header file(myString.h)**

#pragma once

#include<string>

using namespace std;

class myString {

private:

char \*str;

public:

myString();

void setString(string s); //– assign string to myString data value

string getString(); // returns string of data from myString data value

int size(); //– returns how many characters are in the string(empty string is size zero)

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

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

char\* partString(int startPos, int len); // returns string from startPos for length given

//(handle startPos < 0, startPos = size returns null string, handle startPos > size)

char\* replPartString(myString s, int startPos); //replaces characters starting at startPos with input string

void replWholeString( myString st); // replaces current string data value with input parameter string

bool compareString(myString st); //– compare current value of string with input parameter string

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

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

bool numericString(); //– returns Boolean telling if data value is either integer or real(signs, decimal point.etc.)

bool alphabeticString(); //– returns Boolean telling if data value is all alphabetic characters

};

**Implementation File:**

#include "myString.h"

#include<iostream>

using namespace std;

const int MAX = 25;

myString::myString(){

str = new char[MAX];

}

void myString::setString(string s)

{

if (s.length() > MAX) {

cout << "Error!!Too many characters from input." << endl;

}

else {

int x;

for (x = 0; s[x] != '\0'; x++) // Loops till end of the implicit object string

str[x] = s[x]; // Assigns each character of parameter string st to implicit object string str

str[x] = '\0'; // Assigns null character

}

}

string myString::getString()

{

return str;

}

int myString::size()

{

int len;

for (len = 0; str[len] != '\0'; len++);

return len;

}

void myString::addStart(myString st)

{

char \*temp = new char[st.size() + size()];

if (st.size() + size() > 25)

cout << "\n ERROR: Unable to add at the beginning: exceeds length 25.";

else

{

int x;

for (x = 0; st.str[x] != '\0'; x++)

// Assigns each character of the parameter object of the string to temp

temp[x] = st.str[x];

temp[x++] = ' ';

for (int y = 0; str[y] != '\0'; y++, x++)

// Assigns each character of the implicit object of the string to temp

temp[x] = str[y];

temp[x] = '\0'; // Assigns null character

str = temp; // Assigns concatenated string temp to implicit object string

}

}

void myString::addEnd(myString st)

{

// Creates a temporary string and assigns null

char \*temp = new char[st.size() + size()];

if (st.size() + size() > 25)

cout << " ERROR: Unable to add at the end: exceeds length 25.";

else

{

int x;

// Loops till end of the implicit object string

for (x = 0; str[x] != '\0'; x++)

// Assigns each character of the implicit object of the string to temp

temp[x] = str[x];

// Assigns space

temp[x++] = ' ';

// Loops till end of the parameter object string

for (int y = 0; st.str[y] != '\0'; y++, x++)

// Assigns each character of the parameter object of the string to temp

temp[x] = st.str[y];

// Assigns null character

temp[x] = '\0';

// Assigns concatenated string temp to implicit object string

str = temp;

}// End of else

}// End of function

// Function to validate and return string from startPos to end of the implicit object string

char\* myString::partString(int startPos, int len)

{

// Creates a temporary char type pointer and initialize with a char array

char \*temp = new char[size()];

if (startPos < 0) // Checks if first parameter startPos value is negative then display error message

{

cout << " ERROR: Starting position cannot be negative: ";

}

else if (startPos == size())

{

if (size() == 0) {

cout << "Empty string." << endl;

}

else

cout << " Starting position is equal to size.";

}

else if (startPos > size())

{

cout << " ERROR: Starting position exceeding string length. " << endl;

}

else

{

int x;

// Assigns implicit object string parameters starting position to end to temp string

// Loops till end of the implicit object string

for (x = 0; x < size(); x++)

temp[x] = str[startPos++]; // Assigns character at starting position and increase the startPos by one

temp[x] = '\0'; // Assigns null character

// Returns the substring stored in temp string

return temp;

}// End of else

}// End of function

char\* myString::replPartString(myString s, int startPos) {

char \*temp = new char[size()];

if (startPos < 0) // Checks if first parameter startPos value is negative then display error message

{

cout << " ERROR: Starting position cannot be negative: " << startPos;

} // Otherwise checks if first parameter startPos value is equals to

// implicit object string length then display error message

else if (startPos == size())

{

if (size() == 0) {

cout << "Empty string." << endl;

}

else

cout << " Starting position is equals to size.";

}

else if (startPos > size()) // Otherwise checks if first parameter startPos value is greater than

// implicit object string length then display error message

{

cout << " ERROR: Starting position exceeding string length" << endl;

}

else

{

int x;

for (x = 0; s.str[x] != '\0'; x++)

// Assigns each character of the parameter object of the string to temp

temp[x] = s.str[x];

for (int y = startPos; str[y] != '\0'; y++, x++)

// Assigns each character of the implicit object of the string to temp

temp[x] = str[y];

temp[x] = '\0';

str = temp;

}

return str;

}

// Function to replace implicit object string with parameter object string

void myString::replWholeString(myString st)

{

str = st.str;

}// End of function

// Function to return true if implicit object string is equals to parameter object string

// Otherwise returns false

bool myString::compareString(myString st)

{

// Checks if implicit object string length

// is not equals to parameter object string length then return false

if (size() != st.size())

return false;

// Otherwise size is equal

else

{

// Loops till end of the implicit object string

for (int x = 0; x < size(); x++)

// Checks if each character of implicit object string with

// each character of parameter object string

if (str[x] != st.str[x])

// If not equal then return false

return false;

}// End of else

// Otherwise return true

return true;

}// End of function

// Function to reset the implicit object string to null

void myString::initString()

{

str = (char\*)"";

}// End of function

void myString::printStringScreen()

{

cout << str;

}// End of function

bool myString::numericString()

{

// Loops till end of the implicit object string

for (int x = 0; x < size(); x++)

// Checks if current index position of the string is not a digit then return false

if (!isdigit(str[x]))

return false;

// At the end of the loop returns true because all the index position

// of the implicit object string contains numbers

return true;

}// End of function

// Function to return true if implicit object string all alphabets

// Otherwise returns false

bool myString::alphabeticString()

{

for (int x = 0; x < size(); x++)

if (isalpha(str[x]))

return true;

return false;

}// End of function

**Main file:**

#include<iostream>

#include<fstream>

#include"myString.h"

using namespace std;

int main() {

myString first, second, third, fourth, fifth, sixth, seventh;;

ofstream outs;

outs.open("results.txt");

first.setString("First");// validate the set function

second.setString("Second");

third.setString("Third");

fourth.setString("");

sixth.setString("thisisalongstring");

seventh.setString("anotherlongstring");

outs << "setString: myString first = First Success" << endl;

outs << "setString: myString second = Second Success" << endl;

outs << "setString: myString third = Third Success" << endl;

outs << "setString: myString fourth = " << endl;

cout << "First string is: " << first.getString() << endl << endl; //validate get methods

outs << "getString: myString first Result: " << first.getString() << endl;

cout << "Second string is: " << second.getString() << endl << endl;

outs << "getString: myString second Result: " << second.getString() << endl;

cout << "Third string is: " << third.getString() << endl << endl;

outs << "getString: myString third Result: " << third.getString() << endl;

cout << "Fourth string is: " << fourth.getString() << endl << endl;

outs << "getString: myString fourth Result: " << fourth.getString() << endl;

fifth.setString("gugisifbehbshdbfakgkjdfjak"); // setting a string of size more than 25

outs << "setString: myString fifth Message: Error!!Too many characters from input." << endl;

cout <<endl<< "Sixth string is: " << sixth.getString() << endl << endl;

outs << "getString: myString sixth Result: " << sixth.getString() << endl;

cout << "Seventh string is: " << seventh.getString() << endl << endl;

outs << "getString: myString seventh Result: " << seventh.getString() << endl;

cout << endl << "The size of the first string is: " << first.size() << endl; //test size() function

outs << "Size(): myString first = "<<first.getString()<<" Result: " << first.size() << " Message: The size of the first string is : " << first.size() << endl;

cout << endl << "The size of the fourth string is: " << fourth.size() << endl; //test size() function

outs << "Size(): myString first = " << fourth.getString() << " Result: " << fourth.size() << "Message: The size of the fourth string is: " << fourth.size() << endl;

cout << "----------------Adding first string infront of second string---------------------- "<<endl;

second.addStart(first); // test the addStart method functionalities

cout << "After operation, the new second string is: " << second.getString() << endl << endl;

outs << "addStart() myString second = Second Parameter: first Success: After adding, the new second string is: " << second.getString()<< endl;

cout << "----------------Adding sixth string infront of seventh string---------------------- " << endl;

seventh.addStart(sixth); // test the addStart method functionalities

cout <<endl<< "After operation, the new seventh string is: " << seventh.getString() << endl << endl;

outs << "addStart() myString seventh = " << seventh.getString() << " Parameter: sixth myString ERROR: Unable to add at the beginning: exceeds length 25" << endl;

cout << "---------------Adding third string at the end of first string----------------------" << endl;

first.addEnd(third); // test the addEnd method functionalities

cout << "After operation, the new first string is: " << first.getString() << endl << endl;

outs << "addEnd() myString first = First Parameter: third Success: After adding, the new first string is: " << first.getString() << endl;

cout << "----------------Adding sixth string at the end of seventh string---------------------- " << endl;

seventh.addStart(sixth); // test the addStart method functionalities

cout << endl << "After operation, the new seventh string is: " << seventh.getString() << endl << endl;

outs << "addEnd() myString seventh = " << seventh.getString() << " Parameter: sixth myyString ERROR: Unable to add at the beginning: exceeds length 25" << endl;

cout <<endl << "---------------Returning first string from position -1 --------------------------- " << endl;

first.partString(-1, first.size()); // validate for negative value for starting position

outs << "partString(): myString first = " << first.getString() <<" Parameter: -1, first.size() ERROR: Starting position cannot be negative" << endl;

cout<< endl << "-----------------Returning first string from position 15:--------------------------- "<< endl;

first.partString(15, first.size()); // testing for starting point exceeding the string length

outs << "partString(): myString first = " << first.getString() << " Parameter: 15, first.size() ERROR: Starting position exceeding string length." << endl;

cout << endl << "-------------Returning fourth string from position of size(NULL): -------------------" << endl;

fourth.partString(fourth.size(), fourth.size()); // testing for when the starting point size is NULL

outs << "partString(): myString fourth = " << fourth.getString() << " Parameter: fourth.size(), fourth.size() Message: Empty string." << endl;

cout << endl << "---------Returning first string from position of size(Not Null): ------------ "<< endl;

first.partString(11, first.size()); // testing for when the starting point is equal to string length

cout << endl <<"After returning, the first string is: " << first.getString() << endl;

outs << "partString(): myString fourth = " << fourth.getString() << " Parameter: fourth.size(), fourth.size() Message: ERROR: Starting position exceeding string length.." << endl;

//returning string from a valid starting position

cout << endl << "-------------Returning string from position 2 of first string:---------------" << endl;

cout << "After returning, the first string is: " << first.partString(2, first.size()) << endl;

string s = first.partString(2, first.size());

outs << "partString(): myString first = " << first.getString() << " Parameter: 2, first.size() Success: After returning, the first string is: "<< s << endl;

cout << endl << "---------Replacing character of a string with a parameter string from a negative position--------" << endl;

cout << "Second string: " << second.getString() << " and Third String: " << third.getString() << endl;

second.replPartString(third, -1);

outs << "replPartString(): myString second = " << second.getString() << " Parameter: third, -1 ERROR: Starting position cannot be negative" << endl;

cout << endl << "------Replacing character of a string with input string from position 15 --------" << endl;

cout << "Second string: " << second.getString() << " and Third String: " << third.getString() << endl;

second.replPartString(third, 15);

outs << "replPartString(): myString second = " << second.getString() << " Parameter: third, 15 Message: ERROR: Starting position exceeding string length." << endl;

cout << endl << "---------Replacing character of a string with a parameter string from a position of size(NULL)--------" << endl;

cout << "Fourth string: " << fourth.getString() << " and Third String: " << third.getString() << endl;

fourth.replPartString(third, fourth.size());

outs << "replPartString(): myString fourth = " << fourth.getString() << " Parameter: third, fourth.size() Message: Empty string." << endl;

cout << endl << "---------Replacing character of a string with a parameter string from a valid position--------" << endl;

cout << "Second string: " << second.getString() << " and Third String: " << third.getString() << endl;

cout << "After replacing, the new second string is: " << second.replPartString(third, 2) << endl;

outs << "replPartString(): myString second = " << second.getString() << " Parameter: third, 2 Success: After replacing, the second string is: " << second.replPartString(third, 2) << endl;

cout<< endl << "First string: " << first.getString() << " and Third string: " << third.getString() << endl;

cout << "--------------Replace first string with third string----------------"<< endl;

first.replWholeString(third);

cout << endl << "After replacing, the new first string is: " << first.getString() << endl;

outs << "replWholeString(third) myString first = First Third Result: third Success: After replacing, the new first string is : " << first.getString() << endl;

cout << endl << "---------------Comparing two string values---------------------" << endl;

cout << endl << "First string: " << first.getString() << " and Third string: " << third.getString() << endl;

cout << endl << "comparing first and third string: " << endl;

if (first.compareString(third) == false) {

cout << "Not Equal." << endl;

outs << "compareString(third): myString first = " << first.getString() << " Parameter: third myString Message: Not Equal. " << endl;

}

else {

cout << "Both are equal." << endl << endl;

outs << "compareString(third): myString first = " << first.getString() << " Parameter: third myString Message: Both are equal" << endl;

}

cout << endl << "First string: " << first.getString() << " and Second string: " << second.getString() << endl;

cout << endl << "Comparing first and second string: " << endl;

if (first.compareString(second) == false) {

cout << "Not Equal." << endl;

outs << "compareString(second): myString first = " << first.getString() << " Parameter: second myString Message: Not Equal. " << endl;

}

else {

cout << "Both are equal." << endl;

outs << "compareString(second): myString first = " << first.getString() << " Parameter: second myString Message: Both are Equal. " << endl;

}

cout << endl << "---------Resetting the Value to Null-------------- " << endl;

cout << endl << "First String: " << first.getString() << endl;

first.initString(); //resetting first to its null value

cout << endl<< "After resetting to its null value, First String is: " << first.getString() << endl;

outs << "initString(): myString first = Third Result: NULL Success: After resetting to its null value, First String is: " << first.getString() << endl;

cout << endl << "---------Printing Current Third String value-------------- " << endl;

cout << "Third String: ";

third.printStringScreen(); //printing third string in the display

cout << endl;

outs << "printStringScreen() myString third = " << third.getString() << " Result: Third Success: Third String: Third" << endl;

cout << endl << "-----Check if the data is integer or real-------------" << endl;

cout << "Third String: " << third.getString() << endl;

if (third.numericString() == false) {

cout << "Data value has no integer or real." << endl;

outs << "numericString() myString third: " << third.getString() << " Result: False Message: Data value has no integer or real." << endl;

}

else {

cout << "Data value has integer or real." << endl;

outs << "numericString() myString third: " << third.getString() << " Result: True Message: Data value has integer or real." << endl;

}

second.setString("7234");

cout << "Second String: " << second.getString() << endl;

if (second.numericString() == false) {

cout << "Data value has no integer or real." << endl;

outs << "numericString() myString secobd: " << second.getString() << " Result: False Message: Data value has no integer or real." << endl;

}

else

{

cout << "Data value has integer or real." << endl;

outs << "numericString(): myString second: " << second.getString() << " Result: True Message: Data value has integer or real." << endl;

}

cout << endl << "---------------Check if the data has all alphabets-------------" << endl;

cout << endl << "Third String: " << third.getString() << endl;

if (third.alphabeticString() == false) {

cout << "Data value does not have all alphabets." << endl;

outs << "alphabeticString(): myString third: " << third.getString() << " Result: False Message: Data value does not have all alphabets." << endl;

}

else {

cout << "Data value contains all alphabets." << endl;

outs << "alphabeticString(): myString third: " << third.getString() << " Result: True Message: Data value contains all alphabets.." << endl;

}

cout <<endl << "Second String: " << second.getString() << endl;

if (second.alphabeticString() == false) {

cout << "Data value does not have all alphabets." << endl;

outs << "alphabeticString(): myString second: " << second.getString() << " Result: False Message: Data value does not have all alphabets." << endl;

}

else {

cout << "Data value contains all alphabets." << endl;

outs << "alphabeticString(): myString second: " << second.getString() << " Result: True Message: Data value contains all alphabets.." << endl;

}

system("pause");

return 0;

}//

1. **Revised Algorithm**

**Header File:**

//class myString definition

//private

//declaration of ~~array~~ a pointer of character – ~~char a[];~~ char \*str

//public:

//default constructor myString()declaration

// void set method declaration with a string parameter

//string type get method declaration with no parameter

// int type size() function declaration

//void addStart(myString) function declaration

// void addEnd(myString) function declaration

//~~void~~ char\* partString(startPos, length) function declaration

//~~void~~ char\* replPartString(myString, startPos) function declaration

//void replWholeString( myString) function declaration

// boolean type compareString(myString) function declaration

// void initString() function declaration

// void printStringScreen() func

tion declaration

// boolean type of numericString() function declaration

// boolean type of alphabeticString() function declaration

**Implementation File:**

//include header file

//include iostream

//constant variable size of type int MAX declared with the value of 25

// myString() {

~~//initialize the array of character to it’s null position~~

//allocate new memory of type char array with the MAX size to the pointer \*str

}//end of function

// void setString(string s) {

// if the size of string s is greater than 25, show error message

//otherwise

{

// declare variable x

//for loop x = 0 to ~~less than~~ till the end of ~~length of~~ s

// store each character of string s to ~~char a[]~~ object string str

} // end of function

// string getString() {

// return ~~s~~ str;

} //end of function

// int size() {

~~//declare and initialize variable size equal to zero~~

// declare len for length;

// for loop ~~i~~  len= 0 to ~~less then Max~~  until the end of string

{

~~// if character array a[i] is not null, increment size by 1~~

//return length as len;

}

~~//return size;~~

}//end of function

//void addStart(myString st) {

//this function adds the string in the input parameter to the front of current string

// Creates a temporary string by a char pointer

//allocate a new memory for the array with size of st + implicit string

//check if the size of st + size of current string is more than 25

// show an error message

//otherwise, when total size is less than 25

~~// create a new array of char with the new total size~~

// declare int variable x;

// for loop x = 0 to ~~size of the char~~  till the end of the parameter object string

// assign each character of the parameter object of the string to the temporary string

//assign space

// loops till the end of implicit string

// assign each character of current string to the temporary string

// Assigns null character

// Assigns concatenated string to implicit object string

}// end of the function

// void addEnd(myString st) {

// this function adds the string in the input parameter to the end of the current string

// Creates a temporary string by a char pointer

//allocate a new memory for the array with size of st + implicit string

//check if the size of st + size of current string is more than 25

// show an error message

// otherwise, when total size is less than 25

~~// create a new array of char with the new total size~~

// declare int variable x;

// for loop x = 0 ~~to size of the char~~ till end of the implicit object string

// assign each character of current string to the ~~new array of char in the position of x~~ to the temporary string

//assign space

// for loop x = 0 ~~to size of new parameter string~~ till the end of parameter object string

// assign each character of parameter string ~~after the current string characters~~ to temporary string

// Assigns null character

// Assigns concatenated string to implicit object string

} // end of the function

//~~void~~ char\* partString(int startPos, int length) {

//this function returns string from startPos for length given

//(handle startPos < 0, startPos = size returns null string, handle startPos > size)

// Creates a temporary char type pointer and initialize with a char array

// if startPos is negative, show an error message

// else if startPos is equal to size of a string {

// if string is empty, shows a message

// otherwise, shows another message saying starting point and size is equal

}

// else if startPos is greater than the length, show an error message

// otherwise, for loop from x = startPos to length

~~// print each character of the string from the position of x~~

// Assigns character at starting position and increase the startPos by one

// Assigns null character

// Returns the substring stored in temp string

} // end of function

//~~void~~ char\* replPartString(myString s, int startPos) {

//this function replaces characters starting at startPos with input string

//copy and assign input string to the starting position of the current string

}// end of function

//void replWholeString( myString st)

{

// this function replaces current string data value with input parameter string

//copy and assign the parameter string to the current string

}//end of function

// boolean type compareString(myString st) {

// if the size of input string and current is not equal

//return false

// otherwise when the sizes are equal:

~~//if each character of input and current string is same~~

// Loops till end of the implicit object string

// Checks if each character of implicit object string with

// each character of parameter object string

// If not equal then return false

//end of else

// otherwise return true

~~//return true~~

~~// otherwise, return false~~

}//end of function

// void initString() {

// this function resets / initializes string to null string

// assign current string(array of character) is equal to its NULL position

}//end of function

// void printStringScreen() {  
 //this function prints myString data value to the monitor

~~//for int i = 0 to the size of myString array size~~

~~// print each character of array~~

// print the dereferenced value of char pointer

}//end of function

//bool numericString() {

// returns Boolean telling if data value is either integer or real(signs, decimal point.etc.)

// For loops till end of the implicit object string

// Checks if current index position of the string is not a digit

//return false

//otherwise return true

}//end of function

//bool alphabeticString() {

//– returns Boolean telling if data value is all alphabetic characters

// for x = 0 to the size of the array {

// if element of array at the position x is equal to isalpha, return true

// otherwise return false

}

}//end of function

**Main file:**

//include iostream

//include header file

//include fstream library

//using namespace std

// main function () {

// declare 3 object variables – first, second, third, fourth, fifth, sixth, seventh of myString class

// declare ofstream file variable and open an output file to write

// use ofstream file to store information after each action of function call to store it in the file

//set values for each of ~~3~~ 7 objects using set method including

// one with Null, two with values that add up to more than 25 in size

// print the values of each objects using get method

// call the function size() to find the size of a variable(not Null) and print it

// call the function size() to find the size of a variable (Null) and print it

// add the first string infront of the second string to get a new second myString object using second.addStart(first) function call

// add the sixth string infront of the seventh string to get a new seventh myString value using seventh.addStart(sixth) function call

// add the third string at the end of the first string and get a new value for first myString using first.addEnd(third) function call

// add the sixth string at the end of the seventh string and get a new value for seventh myString using first.addEnd(third) function call

//prompt to return the first string from a negative position by calling partString() function

//prompt to return the first string from a position which is greater than the string size using partString() function

//prompt to return the fourth string from a position of fourth string size when fourth string is NULL by calling partString() function

// prompt to return the first string from a position which is equal to the size of the first string by using partString() function and the size is not zero

//prompt to return the first string from position 2 by calling partString() function

// replPartString(myString, startPos) call this function to check different condition while replacing characters with a parameter string

// replace the whole first string object with the third one and get a new first string value by calling replWholeString( myString) with passing the third myString object as a parameter

//compare the value of current first string object = “third” with third string object = “third” by calling Boolean compareString(myString) function while passing third myString object as a parameter

//this should return true and thus print a message

// compare the value of current first string object = “third” with second string object = “first second” by calling Boolean compareString(myString) function while passing second myString object as a parameter

//this should return false and thus print an appropriate message

//reset the value of first string object = “third” to its NULL value by calling initString() function for first string object

//print the current third string object value by calling printStringScreen() function for third string object

//To check if the third string object = “third” has any digit or real, call Boolean numericString()

//this should return false and display an appropriate message.

//set the ~~third~~ second string object with a value = “7234” by using setString method that will update the value of ~~third~~ second string object

//To check if the current ~~third~~ second string object = ~~“third”~~ “7234” has any digit or real, call Boolean numericString()

//this should return ~~false~~ true and thus show another appropriate message in the display

//To check if the third myString object = ~~“7234”~~ “third” has all the alphabetic value, call boolean alphabeticString() function

// this should return ~~false~~ true and thus show an appropriate message

//To check if the second myString object = ~~“first second”~~ “7234” has all the alphabetic value, call boolean alphabeticString() function

// this should return ~~true~~ false and thus show another appropriate message

**//** system pause

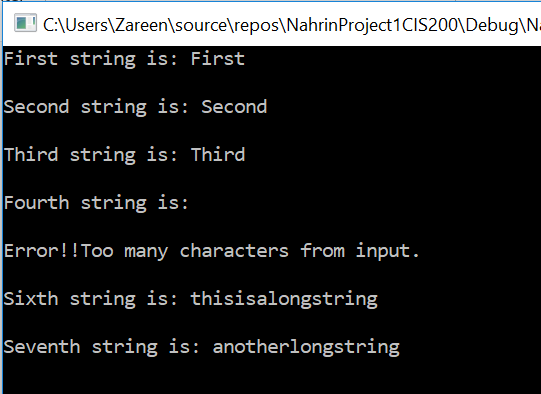
//return 0

//end of main function

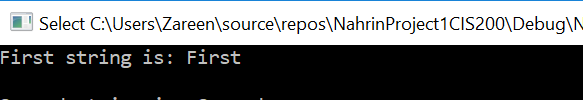
1. **Completed test plan:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | Use set method to insert value in one of the object of class myString  Set a string that has a size less than 25  To check, use get method to check the value for that object and print it. | first.setString(“First”);  first.getString(); | First string is: First | First string is: First | Pass |
| 2 | Invalid | Use set method to insert value in one of the object of class myString  Set a string that has a size more than 25 | Fifth.setString(“gugisifbehbshdbfakgkjdfjak”) | Error!! Too many characters from input | Error!! Too many characters from input | Pass |
| 3 | Valid | Use size function to get the size of a string and print the size | myString first = “first”  first.size(); | The size of the first string is: 5 | The size of the first string is: 5 | Pass |
| 4 | Valid | Use size function to get the size of a NULL string and print the size | myString fourth = “”  fourth.size() | The size of the fourth string is: 0 | The size of the fourth string is: 0 | Pass |
| 5 | Valid | Add first string in front of second string  The sum of the sizes of both strings is less than 25 | myString first = “First”  myString second = “Second” | After operation, the new second string is = First Second | After adding, the new second string is = First Second | Pass |
| 6 | Invalid | Add sixth string in front of seventh string  The sum of the sizes of both strings is more than 25. So the seventh string should not be updated | myString sixth =  “thisisalongstring”  myString seventh =  “anotherlongstring” | ERROR: Unable to add at the beginning: exceeds length 25.  After operation, the new seventh string is: anotherlongstring | ERROR: Unable to add at the beginning: exceeds length 25.  After operation, the new seventh string is: anotherlongstring | Pass |
| 7 | Valid | Add third string at the end of first string  The sum of the sizes of both strings is less than 25 | myString first = “First”  myString third = “Third” | After operation, the new first string is = First Third | After operation, the new first string is = First Third | Pass |
| 8 | Invalid | Add sixth string at the end of seventh string  The sum of the sizes of both strings is more than 25. So the seventh string should not be updated | myString sixth =  “thisisalongstring”  myString seventh =  “anotherlongstring” | ERROR: Unable to add at the beginning: exceeds length 25.  After operation, the new seventh string is: anotherlongstring | ERROR: Unable to add at the beginning: exceeds length 25.  After operation, the new seventh string is: anotherlongstring | Pass |
| 9 | Invalid | Return the first string value from a negative starting position | myString first = “First Third”  position = -1 | Error: Starting point cannot be negative. | Error!! Starting point cannot be negative. | Pass |
| 10 | Invalid | Return the first string value from a starting position which is more than string length | myString first = “First Third”  position = 15 | ERROR: Starting position exceeding string length | ERROR: Starting position exceeding string length | Pass |
| 11 | Valid | Returning a string from a starting position which is equal to the size of the string  The string is NULL | myString fourth = “”  position = fourth.size() | Empty String | Empty String | Pass |
| 12 | Valid | Returning a string from a starting position which is equal to the size of the string  The string is not NULL | myString first = “First Third”  Position = 11 | Starting point is equal to size.  After returning, the first string is: “First Third” | Starting point is equal to size.  After returning, the first string is: First Third | Pass |
| 13 | Valid | Returning a string from a valid starting position | myString first = “First Third”  Position = 2 | After returning, the first string is: “rst Third” | After returning, the first string is: “rst Third” | Pass |
| 14 | Valid | Replace the whole first string with third string and get a new first string | myString first = “First Third” | After replacing, the new first string is: “Third” | After replacing, the new first string is: “Third” | Pass |
| 15 | Invalid | Replace the character of second string from a negative position with the parameter string | second = “First Second”  startPos = -1  third = “Third” | Error: Starting point cannot be negative | Error: Starting point cannot be negative | Pass |
| 16 | Invalid | Replace the character of second string from a position greater than the implicit string with the parameter string | second = “First Second”  startPos = 15  third = “Third” | Error: Starting position exceeding string length | Error: Starting position exceeding string length | Pass |
| 17 | Invalid | Replace the character of fourth string from the position of its size(Null)with the parameter string | fourth = “”  startPos = fourth.size()  third = “Third” | Empty string | Empty string | Pass |
| 18 | Valid | Replace the character of second string from a valid position with the parameter string | second = “First Second”  startPos = 2  third = “Third” | After replacing, the new second string is: Thirdrst Second | After replacing, the new second string is: Thirdrst Second | Pass |
| 19 | Valid | Compare the value of current myString first and myString third | myString first = “Third”  myString third = “Third” | Both are equal | Both are Equal | Pass |
| 20 | Valid | Compare the value of current myString first and myString second | myString first = “Third”  myString second= “First Second” | Not equal | Not equal | Pass |
| 21 | Valid | Reset a string to its Null value | myString first = “Third” | After resetting to its null value, First string is: | After resetting to its null value, First string is: | Pass |
| 22 | Valid | Print the current value of a string using printStringScreen() | myString third = “Third” | Third String = Third | Third String = Third | Pass |
| 23 | Valid | Use numericString() to check if the string has digits or real | myString third = “Third” | Data value has no integer | Data value has no integer | Pass |
| 24 | Valid | Use numericString() to check if the string has digits or real | myString second = “7234” | Data value has integer or real | Data value has integer or real | Pass |
| 25 | Valid | Use alphabeticString() too check if a string has all alphabets. | myString third = “Third” | Data value contains all alphabets. | Data value contains all alphabets. | Pass |
| 26 | Valid | Use alphabeticString() too check if a string has all alphabets. | myString second = “7234” | Data value does not have all alphabets | Data value does not have all alphabets | Pass |

1. **Test Result:**



Test Case 1:



Test Case 2:



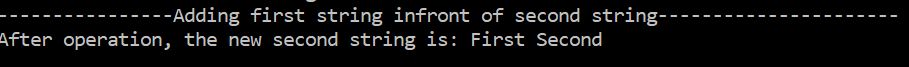
Test Case 3:



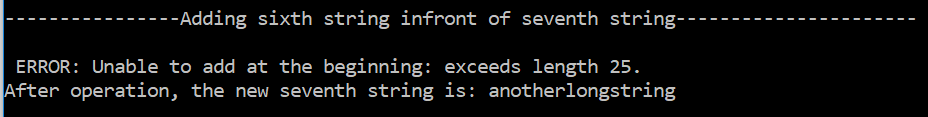
Test Case 4:



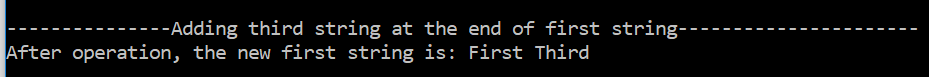
Test Case 5:



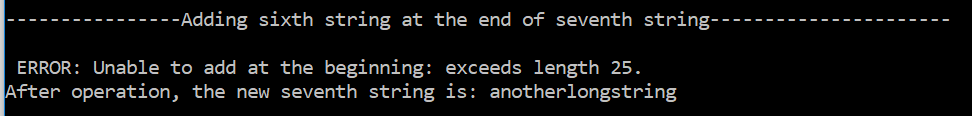
Test Case 6:



Test Case 7:



Test Case 8:



Test Case 9:



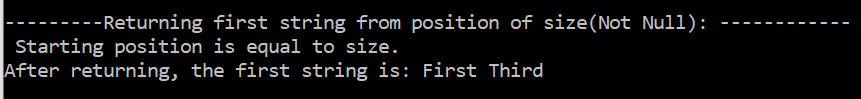
Test Case 10:



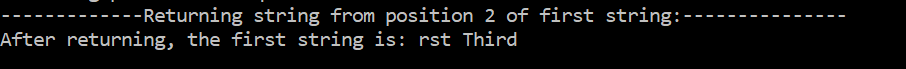
Test Case 11:

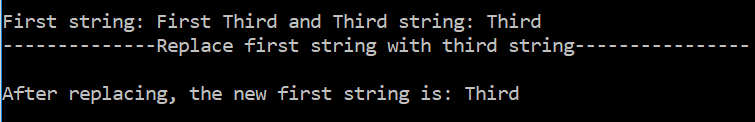


Test Case 12

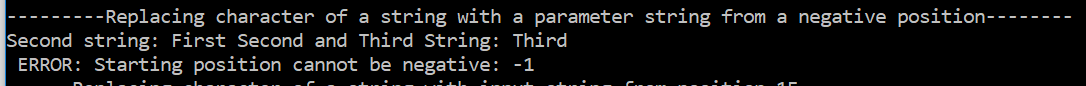


Test Case 13:

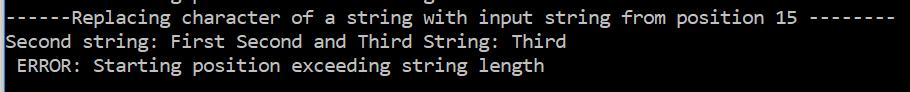


Test Case 14: 

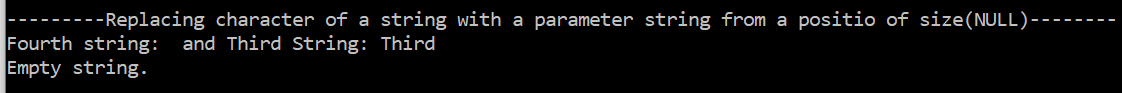
Test Case 15:



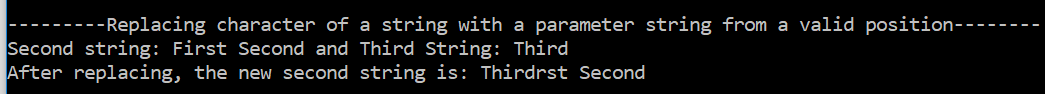
Test Case 16:



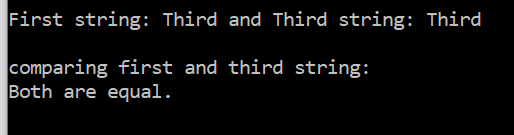
Test Case 17:



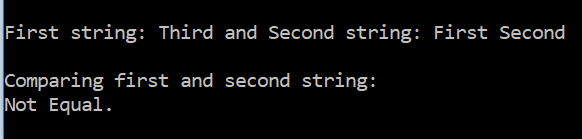
Test Case 18:



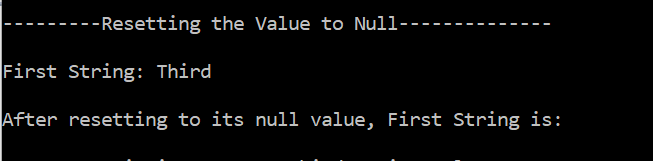
Test Case 19:



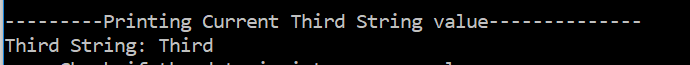
Test Case 20:



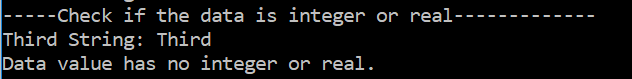
Test Case 21:



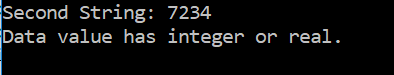
Test Case 22:



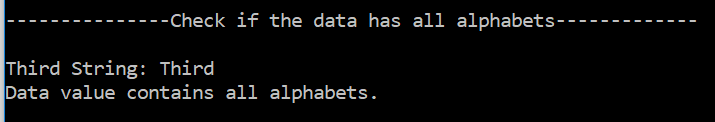
Test Case 23:



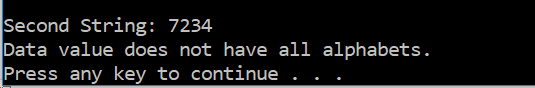
Test Case 24:



Test Case 25:



Test case 26:



1. **Output Files: (“results.txt”)**

**For set and get methods:**

setString: myString first = First Success

setString: myString second = Second Success

setString: myString third = Third Success

setString: myString fourth =

getString: myString first Result: First

getString: myString second Result: Second

getString: myString third Result: Third

getString: myString fourth Result:

setString: myString fifth Message: Error!!Too many characters from input.

getString: myString sixth Result: thisisalongstring

getString: myString seventh Result: anotherlongstring

**For size method:**

Size(): myString first = First Result: 5 Message: The size of the first string is : 5

Size(): myString first = Result: 0Message: The size of the fourth string is: 0

**For addStart method:**

addStart() myString second = Second Parameter: first Success: After adding, the new second string is: First Second

addStart() myString seventh = anotherlongstring Parameter: sixth myString ERROR: Unable to add at the beginning: exceeds length 25

**For addEnd method:**

addEnd() myString first = First Parameter: third Success: After adding, the new first string is: First Third

addEnd() myString seventh = anotherlongstring Parameter: sixth myyString ERROR: Unable to add at the beginning: exceeds length 25

**For partString method:**

partString(): myString first = First Third Parameter: -1, first.size() ERROR: Starting position cannot be negative

partString(): myString first = First Third Parameter: 15, first.size() ERROR: Starting position exceeding string length.

partString(): myString fourth = Parameter: fourth.size(), fourth.size() Message: Empty string.

partString(): myString fourth = Parameter: fourth.size(), fourth.size() Message: ERROR: Starting position exceeding string length..

partString(): myString first = First Third Parameter: 2, first.size() Success: After returning, the first string is: rst Third

**For replPartString method:**

replPartString(): myString second = First Second Parameter: third, -1 ERROR: Starting position cannot be negative

replPartString(): myString second = First Second Parameter: third, 15 Message: ERROR: Starting position exceeding string length.

replPartString(): myString fourth = Parameter: third, fourth.size() Message: Empty string.

replPartString(): myString second = Thirdirdrst Second Parameter: third, 2 Success: After replacing, the second string is: Thirdirdrst Second

**For replWholeString method:**

replWholeString(third) myString first = First Third Result: third Success: After replacing, the new first string is : Third

**For compareString method:**

compareString(third): myString first = Third Parameter: third myString Message: Both are equal

compareString(second): myString first = Third Parameter: second myString Message: Not Equal.

**For initString method:**

initString(): myString first = Third Result: NULL Success: After resetting to its null value, First String is:

**For printStringScreen method:**

printStringScreen() myString third = Third Result: Third Success: Third String: Third

**For numericString method:**

numericString() myString third: Third Result: False Message: Data value has no integer or real.

numericString(): myString second: 7234 Result: True Message: Data value has integer or real.

**For alphabeticString method:**

alphabeticString(): myString third: Third Result: True Message: Data value contains all alphabets..

alphabeticString(): myString second: 7234 Result: False Message: Data value does not have all alphabets.

1. **Error log:**

numericString() might only verify the existence of digit, not other signs.

What should numericString() return if the data value has a combination of digits and other

characters?

1. **Status:**

To store the information in the output file, a function should have been used instead of writing through ofstream variable each time.