You will work on this assignment in teams of 2. You can decide which methods you will write and which methods your partner will write. One of you will put your methods into String1.s and the other will put his/her methods into String2.s I have given you the UML for each and an explanation of what the file does. Understand that at any time the “type” is String, it will mean that the **address** of that string gets passed. I will use both of following formats – just out of habit. In some of the methods you will need dynamic memory allocation method which we have to research how to do in class. All of your methods will be in an external file (String1.o or String2.o) which will be linked to your proj3.obj file, they will all need to be declared “externally”.

Remember to adhere to AAPCS convention. All returned values are in R0. All reference variables are returned in R0.

+String\_length(string1:String):int is identical to +String\_length(lpString1:word):word

This method accepts the address of a string and counts the characters in the string, excluding the NULL character and returns that value as an int (word) in the R0 register.

+String\_equals(string1:String,string2:String):boolean is identical to

+String\_equals(lpString1:word,lpString2:word):byte

This method makes an exact comparison of individual characters in two strings. If any character in the string in a position is different than the character in the same position in the other string, the method returns “false” (0 in the R0 register). If the length of the two strings is different, the method also returns “false”. Note that ‘e’ is NOT the same as ‘E’. Otherwise “true” (1) is returned. The value is returned in the R0 register.

+String\_equalsIgnoreCase(string1:String,string2:String):boolean is identical to

+String\_equalsIgnoreCase(lpString1:word,lpString2:word):byte

This method makes a comparison of individual characters in two strings ignoring case. If any character in the string in a position is different than the character in the same position in the other string, the method returns “false” (0 in the R0 register). If the length of the two strings is different, the method also returns “false”. Note that ‘e’ is the SAME as ‘E’. The value returned is in the R0 register.

+String\_copy(string1:String):String => +String\_copy(lpStringToCopy:word):word

This method accepts a string to copy, allocates dynamically enough storage to hold a copy of the new characters, copies the characters and returns the address of that newly created string. You will need to CALL the alloc method, the UML for which is below. I have written this method: you only have to PROTOtype it and CALL it.

+String\_substring\_1(string1:String,beginIndex:int,endIndex:int):String  
 This method creates a new string consisting of characters from a substring of the passed string starting with beginIndex and ending with endIndex.

+String\_substring\_2(string1:String,beginIndex:int):String  
This method creates a new string consisting of characters from a substring of the passed string starting with beginIndex to the end of the original string.

+String\_charAt(string1:String,position:int):char => +stringCharAt(lpString:word, position:word):byte  
This method returns the character in the indicated position. If the request is impossible to fulfill, the method returns 0

+String\_startsWith\_1(string1:String,strPrefix:String, pos:int):boolean  
 It checks whether the substring (starting from the specified offset index) exists within string1. For example testing the string “George Washington” for the prefix “Wash” starting in position 7 would return “true” (1) otherwise, it would return false (0) would have is having the specified prefix or not.

[+String\_startsWith\_2(string1:String, strPrefix:String)](http://beginnersbook.com/2013/12/java-string-startswith-method-example/):boolean It tests whether string1 begins with the specified prefix. If yes then it returns true else false.

[+String\_endsWith(string1:String, suffix:String):boolean](http://beginnersbook.com/2013/12/java-string-endswith-method-example/) Checks whether the string ends with the specified suffix.

+String\_indexOf\_1(string1:String,ch:char):int Returns the index of first occurrence of the specified character ch in the string.

+String\_indexOf\_2(string1:String,ch:char,fromIndex:int):int Same as indexOf method however it starts searching in the string from the specified fromIndex.

[+String\_indexOf\_3(string1:String, str:String):int](http://beginnersbook.com/2013/12/java-string-indexof-method-example/) This method returns the index of first occurrence of specified substring str.

[+String\_lastIndexOf\_1(string1:String, ch:char)](http://beginnersbook.com/2013/12/java-string-lastindexof-method-example/):int It returns the last occurrence of the character ch in the string.

[+String\_lastIndexOf\_2(string1:String,ch:char,fromIndex:int):int](http://beginnersbook.com/2013/12/java-string-lastindexof-method-example/)  Same as lastIndexOf\_1 method, but it starts search from fromIndex.

+[String\_lastIndexOf\_3(string1:String,str:String)](http://beginnersbook.com/2013/12/java-string-lastindexof-method-example/):int Returns the index of last occurrence of string str.

+[String\_concat(string1:String,str:String)](http://beginnersbook.com/2013/12/java-string-concat-method-example/):String Concatenates the specified string “str” at the end of the string.

+[String\_replace(string1:String,oldChar:char,newChar:char)](http://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/):String It returns the new updated string after changing all the occurrences of oldChar with the newChar.

+[String\_toLowerCase(string1:String):String](http://beginnersbook.com/2013/12/java-string-tolowercase-method-example/) It converts the string to lower case string  
  
+String\_toUpperCase(string1:String):String It converts the string to upper case string

Driver: **RASM3.asm**

Below is a partial list of things you should do to verify your program is working correctly. MY suggestion – let your partner test YOUR methods – and you test his/hers. The list below is NOT a complete list of things you can do to test your methods. When you submit your assignment, both of you submit both rasm3.s and String1.s and String2.s

Input 4 strings, each of at least 10 characters, the last two of which are identical in content, but different in “case”. str1, str2, str3, str4

1. Determine the length of the first string and post an appropriate message
2. Determine if str1 and str2 are equal and post an appropriate message
3. Determine if str3 and str4 are equal and post an appropriate message
4. determine if str3 and str4 are equal ignoring the case and post an appropriate message
5. create a 5th string, str5 that is a copy of str1
6. Determine if str1 and str5 are equal and post an appropriate message
7. Create a 6th string, str6 and is a substring of str1 that begins in position 1 and ends in position 5 and post an appropriate message
8. Use str6 again to contain the address of a substring of str1 that begins just in position 1 and post an appropriate message
9. \*\*Input **iPosition** and test the
10. Display the single character at position 3 in str1 and post an appropriate message
11. \*\*Input strTest and input **iPosition**  (converted asci value). Determine if str1 starts with strTest starting in **iPosition** and post an appropriate message. Do this twice testing your startsWith method so that it tests both true and false
12. \*\*Input strTest
13. Determine if str1 starts with strTest and post an appropriate message
14. \*\*Input strTest
15. Determine if str1 ends with strTest and post an appropriate message  
    etc.