תרגיל בית 4 - ADT & Modules

הגשה ביחידים

תרגיל זה יעסוק ב**טיפוסי נתונים מופשטים** (ADT) **ותכנות מודלורי** (Modules) בשפת C.

נושא התרגיל

בשפת C, מחרוזות הן רק מערך של תווים המקודדים ASCII עם התו '\0' בסוף שמסמן את סוף המחרוזת;
ולכן, עד עכשיו השתמשנו בסיפריית string.h כדי להשתמש בפונקציונאליות הנלוות לה.
בתרגיל זה אתם מתבקשים לספק **טיפוס נתונים מופשט (ADT)** חדש ומשופר למחרוזות (**StringADT**)
ובנוסף לממש את הפונקציות המפורטות בהמשך.

מבנה הטיפוס

יש להשתמש במבנה לייצוג מחרוזת.
מבנה זה יכיל מערך שהוקצה באופן דינמי עבור המחרוזת, כמו גם את אורך המחרוזת וגודל המערך.
עליך להשתמש במבנה מחרוזת זה ללא שינוי. שדותיו הם:

- **data:** זהו מצביע למערך תווים, אשר יכיל את תווי המחרוזת שהמבנה שלך מייצג.
- **length:** אורך המחרוזת (מספר התווים המשומשים/תפוסים במערך data), לא כולל '\0'.
- **capacity:** מספר המייצג את גודל המערך data.

על המודול לתמוך בפונקציונאליות המפורטת בסעיף הבא.

תיאור הפונקציות

1. יצירת String:

יצירת String בהינתן מחרוזת תווים(char *), יש להחזיר משתנה מטיפוס String.

בפונקציה זו יש לקבל מחרוזת של C (מערך של תווים), ליצור String על כל המשתמע מכך, להעתיק את המחרוזת לתוך המחרוזת שיצרתם, לעדכן בהתאם את שני השדות הנותרים ולהחזיר את ה-String החדש.

2. יצירת String כמכפל של תת מחרוזת:

יצירת String בהינתן String ומשתנה מטיפוס int, יש להחזיר משתנה מטיפוס String.

בפונקציה זו יש לקבל String (תת המחרוזת) ומספר מטיפוס int שמהווה את המכפלה של תת המחרוזת שקיבלנו, ליצור String על כל המשתמע מכך, להעתיק את המחרוזת לתוך המחרוזת שיצרתם במכפלה של המשתנה ה-int-י שהתקבל, לעדכן בהתאם את שני השדות הנותרים ולהחזיר את ה-String החדש.

לדוגמא: עבור תת מחרוזת "ABC" והמספר 3, המחרוזת של ה-String החדש שיווצר תהיה "

."ABCABCABC

3. הריסת String:

הריסת String בהינתן String.

בפונקציה זו יש לשחרר את ה-String על כל המשתמע מכך. כלומר, לשחרר כל זיכרון דינמי שהוקצה.

4. החזרת אורך המחרוזת (getter function):

החזרת אורך המחרוזת בהינתן String, יש להחזיר משתנה מטיפוס int.

פונקציה זו, בהינתן String, מחזירה את אורך המחרוזת (שנמצאת בתוך ה- data).

5. החזרת המחרוזת (getter function):

החזרת המחרוזת בהינתן String, יש להחזיר מצביע מטיפוס *char.

פונקציה זו, בהינתן String, מחזירה את המחרוזת של ה-String הנתון.

6. שכפול String:

שכפול String בהינתן String, יש להחזיר משתנה מטיפוס String.

פונקציה זו יוצרת עותק חדש של String הניתן ומחזירה אותו.
יש לציין שאחרי יציאה מהפונקציה, כל שינוי במחרוזת של ה-String המקורי לא ישפיע על המחרוזת של ה-String המועתק ולהפך.

7. החזרת תו באינדקס ספציפי (getter function):

החזרת תו באינדקס ספציפי בהינתן String ואינדקס במחרוזת, מטיפוס int, יש להחזיר משתנה מטיפוס char.

פונקציה זו, בהינתן String ואינדקס במחרוזת, מחזירה את התו המצוי באותו האינדקס.

8. חיפוש מיקום תו (אינדקס)(getter function):

חיפוש מיקום תו (אינדקס) בהינתן String ותו ספציפי char, יש להחזיר משתנה מטיפוס int.

פונקציה זו, בהינתן String ותו לחפש אותו במחרוזת, מחזירה את האינדקס של התו, או 1- אם לא ניתן למצוא את התו במחרוזת.
במקרה והתו מופיע מס' פעמים במחרוזת, יש להחזיר את הופעתו הראשונה (התחלה ממיקום 0).

9. בדיקת שוויון בין שני String:

בדיקת שוויון בין שני String בהינתן שני String, יש להחזיר משתנה מטיפוס bool.

פונקציה זו, בהינתן String, מחזירה True במקרה והם שווים ו- False אם לא.
יש לשים לב לבצע השוואת מחרוזות ולא השוואת כתובות.

10. חיבור בין שתי מחרוזות:

חיבור בין שתי מחרוזות בהינתן שני String, יש להחזיר משתנה מטיפוס String.

בהינתן שני String בשמות dest (פרמטר ראשון) ו- src (פרמטר שני), יש להוסיף את src למחרוזת של- dst (מיד אחריה).
יש לזכור שאין לחרוג מהקיבולת של dst, במקרה והקיבולת הפנוייה ב- dst קטנה מגודל המחרוזת של src, יש להגדיל את המחרוזת של dst (כמובן גם לעדכן שדות רלוונטים) ולאחר מכן להוסיף את התווים.
יש להחזיר את dst.

11. העתקת מחרוזת למחרוזת:

העתקת מחרוזת למחרוזת בהינתן שני String, יש להחזיר משתנה מטיפוס String.

בהינתן שני String בשמות dest (פרמטר ראשון) ו- src (פרמטר שני), יש להחליף את התווים ב- dst בתווים מ- src. אין להציף את הקיבולת (capacity) של המחרוזת של dst. אם המחרוזת של src ארוכה יותר מהקיבולת ל- dst, יש להגדיל את הקיבולת של dst (כמובן גם לעדכן שדות רלוונטים) ולאחר מכן להחליף את התווים.
יש לעדכן את אורך ה- dst ולהחזיר את dst.

12. שינוי המחרוזת לאותיות קטנות/גדולות:

שינוי מחרוזת לאותיות גדולות/קטנות בהינתן String ו- פרמטר enum שמסמל אותיות גדולות/קטנות.

פונקציה זו, בהינתן String וסימון של אותיות גדולות/קטנות כפרמטר enum, משנה את המחרוזת בהתאם.
כאשר הפרמטר השני מסמן שיש להחליף לאותיות גדולות, יש להחליף את התווים הקטנים במחרוזת לגדולים (uppercase).
כאשר הפרמטר השני מסמן שיש להחליף לאותיות קטנות, יש להחליף את התווים הגדולים במחרוזת לקטנים (lowercase).

13. בדיקה האם כל התווים במחרוזת הם ספרות:

בודקת האם המחרוזת מכילה אך ורק ספרות בהינתן String, יש להחזיר משתנה מטיפוס bool.

פונקציה זו, בהינתן String, בודקת האם כל התווים במחרוזת הם ספרות (0-9) ומחזירה את התוצאה.
אם כלל התווים הם ספרות הפונקציה מחזירה True, אחרת מחזירה False.

14. מציאת אינדקס של תת מחרוזת:

מוצאת את האינדקס של תת מחרוזת בהינתן String יעד ותת מחרוזת מטיפוס String, יש להחזיר

משתנה מטיפוס int.

פונקציה זו, בהינתן String יעד ותת מחרוזת, מוצאת את האינדקס של תחילת תת המחרוזת שהתקבלה במחרוזת של ה-String (השני).

לדוגמא: עבור תת מחרוזת "ABC" ומחרוזת- "DDDABCDDD", האינדקס שאמור להתקבל הוא 3.

15. מחיקת תת מחרוזת ממחרוזת:

מוחקת תת מחרוזת ממחרוזת בהינתן String יעד ותת מחרוזת מטיפוס String.

פונקציה זו, בהינתן String יעד ותת מחרוזת, במידה ותת המחרוזת קיימת במחרוזת של String, מוחקת את המופע הראשון של תת המחרוזת ממחרוזת ה-String ומצמצמת אותה (שלא יהיו חורים).

16. הדפסת String:

מדפיסה String בהינתן String.

פונקציה זו, בהינתן String, מדפיסה את המחרוזת של ה-String.

17. בונוס- איטרטור ל-String:

יש להגדיר איטרטור למילים במחרוזת ה-String.
האיטרטור יחזיר את המילה הבאה, כאשר תחילת מחרוזת/סוף מחרוזת יכולה להוות תחילת מילה/סוף מילה וגם לאחר רווח (delimiter) אחד/יותר.
יש לשים לב שמילה לא כוללת רווחים!

יש בנוסף גם לספק למשתמש המודול שלכם מאקרו בשם STRING_FOREACH כדי שיוכל לעבור על כל אחת מהמילים שקיימות לו ב-String;
המאקרו ייצר ויחזיר עבור כל אחת מהמילים משתנה מסוג String, יש לזכור שעל אחריות המשתמש לשחרר את ה-String שנוצר בסוף כל איטרציה.

שימו לב, הבונוס מהווה 10 נקודות.

הנחיות

- יש להסיק מן הכתוב את חתימות הפונקציות, ואת אופן מימושן.
- אין להשתמש בספריות אחרות חוץ מ- stdbool.h, stdio.h, stdlib.h, string.h ו- assert.h.
- כמו שניתן להבין בסעיף הקודם- יש להשתמש בספריית assert.h בתוכנית כדי להגן מפני שגיאת תכנות.
- חל איסור להגדיר מבנים אחרים או להגדיר משתנים גלובליים כלשהם.
- יש לזכור- יש לעדכן בהתאם את השדות של String לאחר כל שינוי במחרוזת (data) .
- כל הודעות השגיאה של כלל הפונקציות צריכות להירשם לערוץ ***stderr*** בלבד.

הערות

- ייתכן שתצטרכו לכתוב פונקציות עזר נוספות, אך שימו לב איזה פונקציות אתם צריכים שירשמו ב-interface.
- השתדלו לרשום הערות לפונקציות עזר/ בפרוצדורות מורכבות בקוד שלכם.
- אפשר להגיש את קבצי המימוש: stringADT.c, stringADT.h כקובץ ZIP.
- רצוי לבדוק את כל הפונקציות -- ולא להסתמך רק על תוכנית הבדיקה שמצ'ב לעיל.
- יש לשים לב שבתוכנית הבדיקה לדוגמא רשום לקראת סוף פונקציית ה- ()main שימוש במאקרו *STRING_FOREACH*, מי שלא עושה את הבונוס צריך למחוק את השורות הללו כדי שלא יקבל שגיאה בעת הבדיקה העצמית.
- אין צורך להגיש את תוכנית הבדיקה שלכם.

תוכנית דוגמא לבדיקת המודול:

שימו לב שתוכנית דוגמא זו גם מצורפת לתרגיל כקובץ c.

```
#include "stringADT.h"

int main(){
    int index=3;
    char c='C';
    CASE e_lowercase=LOWER_CASE;
    String s1,s2,s3,s4,s5,s6;

    //Creates new string s1 with given C string
    s1=string_create("ABC");
    //Prints the char string and the length of s1 string
    printf("String s1: %s\n length s1: %d\n\n",string_chars(s1),string_length(s1));

    //Creates new string s2 by given string s3 and the amount of times to copy it
    printf("S1: ");
    string_print(s1);
    printf("S2 after creation by multString func(using mult S1 3 times)->");
    s2=multString_create(s1,3);
    string_print(s2);


    //Creates a new copy s3 of s2
    printf("\nS2: ");
    string_print(s2);
    printf("S3 after creation by duplicating S2-> ");
    s3=string_dup(s2);
    string_print(s3);


    //Prints the char from s3 string for the given index
    printf("\nIn S3: %s the char in index %d is: ", string_chars(s3), index);
    printf("%c\n",string_charByIndex(s3,index));
    //Prints the index from s3 string for the given char
    printf("In S3: %s the first index of char %c is: ", string_chars(s3), c);
```

```
printf("%d\n\n",string_findIndexByChar(s3,c));

//Prints if the string s2,s3 are equal or not
printf("S2: ");
string_print(s2);
printf("S3: ");
string_print(s3);
printf("The Strings S2 and S3 are: %s\n\n",string_isEqual(s2,s3)?"Equal":"Not Equal");

printf("S1: ");
string_print(s1);
printf("S2: ");
string_print(s2);
printf("The Strings S1 and S2 are: %s\n\n", string_isEqual(s1,s2)?"Equal":"Not Equal");

s4=string_create("123abc");
printf("S4: ");
string_print(s4);
//Add s4 to s1
printf("S1: %s after adding the string S4: %s -> new S1: ", string_chars(s1),
        string_chars(s4));
s1=string_addString(s1,s4);
string_print(s1);

//Copy the chars in s4 to s3
printf("\nS3: %s after copying the chars in the string S4: %s -> new S3:
        ",string_chars(s3), string_chars(s4));
s3=string_copy(s3,s4);
string_print(s3);

//Change the UPPER letters to lower letters
printf("\nS3: %s after using convertCase func -> new S3: ", string_chars(s3));
string_convertCase(s3,e_lowercase);
string_print(s3);

s5=string_create("abc");
printf("\nS5: ");
string_print(s5);
//Prints the index of string s5 in the string s3
printf("In S3: %s the index of the first appearance of S5: %s is: ",
        string_chars(s3),string_chars(s5));
printf("%d\n\n",string_findString(s3,s5));

//Remove from s4 the string s5
printf("S4: %s after removing S5: %s -> new S4: ", string_chars(s4),string_chars(s5));
string_removeString(s4,s5);
```

```
    string_print(s4);

    //Prints if the string s4 contains only digits
    printf("\nThe String S4: %s %s all digits\n\n",string_chars(s4),string_isAllDigits(s4)?"is":"is
            not");

    s6=string_create("aaa, bbb. ccc: ddd?");
    printf("S6: ");
    string_print(s6);
    STRING_FOREACH(word,s6){
            printf("word: ");
            string_print(word);
            string_destroy(word);
    }

    string_destroy(s1);    string_destroy(s2);
    string_destroy(s3);    string_destroy(s4);
    string_destroy(s5);    string_destroy(s6);
}
```

פלט ריצה של תוכנית הדוגמא:

```
String s1: ABC
length s1: 3

S1: ABC
S2 after creation by multString func(by mult S1 3 times)-> ABCABCABC

S2: ABCABCABC
S3 after creation by duplicating S2-> ABCABCABC

In S3: ABCABCABC the char in index 3 is: A
In S3: ABCABCABC the first index of char C is: 2

S2: ABCABCABC
S3: ABCABCABC
The Strings S2 and S3 are: Equal

S1: ABC
S2: ABCABCABC
The Strings S1 and S2 are: Not Equal

S4: 123abc
S1: ABC after adding the string S4: 123abc -> new S1: ABC123abc
```

S3: ABCABCABC after copying the chars in the string S4: 123abc -> new S3: 123abcABC

S3: 123abcABC after using convertCase func -> new S3: 123abcabc

S5: abc
In S3: 123abcabc the index of the first appearance of S5: abc is: 3

S4: 123abc after removing S5: abc -> new S4: 123

The String S4: 123 is all digits

S6: aaa, bbb. ccc: ddd?
word: aaa,
word: bbb.
word: ccc:
word: ddd?