שפת C – תרגיל בית 1

היכרות עם השפה: טיפוסי משתנים, קוד ASCII, לולאות, קלט פלט

טרם בואכם לפתור את תרגיל הבית הראשון, אנו ממליצים לכם לקרוא את הנחיות ההגשה והמדיניות הנמצאים באתר הקורס.

בתרגיל זה נתרגל משימות הכוללות קלט ופלט מהמקלדת, טבלת ASCII, המרות מספרים ו-parsing.
לצורך פתרון התרגילים מומלץ כי תשתמשו בקישורים הבאים.

http://www.cplusplus.com/reference/clibrary/cstdio/getchar
http://www.cplusplus.com/reference/clibrary/cstdio/printf/
http://www.cplusplus.com/reference/clibrary/cstdio/scanf/
http://www.asciitable.com/

- בתרגיל זה אין להשתמש במערכים. שימוש במערכים יוביל לפסילת הסעיף הרלוונטי.
- בדקו כי תכניתכם מתקמפלת ללא אזהרות על ידי שימוש בדגל Wall– בפקודת הקומפילציה.
- לצורך בהירות, במהלך הגדרת התרגילים, התו n\ מציין מעבר שורה.

שאלה 1 (12 נקודות)

ענו בקצרה בקובץ ה-README על השאלות הבאות.

1. טיפוסים בשפה (נקודה אחת)

לגבי כל אחד מן הטיפוסים הבאים ענו על השאלות הבאות בקובץ ה README

1. מהי צריכת הזיכרון של טיפוס זה מבחינת מספר הבתים bytes – הניחו כי השאלה מתייחסת למהדרים המותקנים בתחנות האוניברסיטה לתלמידי מדעי המחשב.
2. מהו טווח הייצוג של טיפוס זה: מינימום ומקסימום.
3. תנו דוגמא (מילולית, שורה אחת) למקרה בו הייתם בוחרים להשתמש בטיפוס מסוג זה.

char, unsigned char, int, unsigned int, short, unsigned short,
long, unsigned long, double, float.

2. (נקודה אחת).
התבוננו בקטע הקוד הבא והסבירו בקצרה מה בעייתי בקטע קוד זה.

```
void getKeyboardInputUntilEOF()
{
   char c;

   while ( ( c = getchar() ) != EOF)
   {
       doSomething(c);
    }
}
```

3. (נכון / לא נכון, נימוק קצר. נקודה אחת):
לאחר ביצוע כל אחת משתי הפקודות שלהלן התו c יכיל את אותו ערך.

```
char c = '0';
char c = 48;
```

4. (משפט אחד, נקודה אחת)
מה מבצעות השורות שלהלן? מדוע?

```
#include <stdio.h>

int main()
{
     int x = 0;
     if (x = 1)
          printf("x=1\n");
     return 0;
}
```

5. (בחר את התשובה הנכונה,נמקו, נקודה אחת)
קטע הקוד הבא:

```
int main()
{
   double a = 0;
   double b = 2 / a;
   printf("b = %f\n", b);
   printf("LABC. \n");
   return 0;
}
```

1. לא עובר קומפילציה.
2. יגרום לזריקת שגיאת זמן ריצה.
3. מדפיס את ההודעה LABC, ומדפיס מחרוזת המסמנת אינסוף.
4. מדפיס את ההודעה LABC + לפחות ספרה אחת.

6. (5 נקודות).

נתבונן במטריצת מספרים 3X3

.

A11	A12	A13
A21	A22	A23
A31	A32	A33

המטריצה תוגדר כ"ריבוע קסם חיבורי" באם סכום האיברים בכל שורה עמודה ואלכסון (2 אלכסונים) שווה.

המטריצה תוגדר כ"ריבוע קסם כפלי" באם מכפלת האיברים בכל שורה עמודה ואלכסון (2 אלכסונים) שווה.

עליכם לממש את הפונקציה checkMagicSquare הבודקת האם ריבוע 3X3 הינו ריבוע קסם כפלי או חיבורי. הפונקציה תהיה בעלת החתימה הבאה:

```
int checkMagicSquare(int type, double A11, double A12, double A13, double A21,
double A22, double A23, double A31, double A32, double A33);
```

בקובץ ~/labc/www/ex1/MyMagicSquare.c מודגמת קריאה לדוגמא לפונקצייה שלכם ב - main.
<u>ממשו</u> את הפונקצייה בקובץ זה, וודאו כי הקובץ עובר קומפילציה.

תיאור הפונקצייה:

הפרמטר type מורה על סוג הבדיקה: 1 לבדיקת ריבוע קסם חיבורי, 2 לבדיקת ריבוע קסם כפלי.
הניחו כי ערך הפרמטר אותו תקבלו תקין (1 או 2)

שאר הפרמטרים הינם איברי הריבוע כפי המוצג לעיל (מסוג double)

ערך ההחזרה יהיה 1 באם הריבוע הינו ריבוע קסם (כפלי או חיבורי, לפי הדרישה), אחרת 0.

הסבירו בקובץ ה README אילו ייתרונות היינו משיגים לו היינו משתמשים במערכים כפרמטר לפונקצייה במקום ברשימת האיברים, פרטו לפחות שני ייתרונות.

7. (2 נקודות)

התבוננו בקטע הקוד הבא וענו על השאלות בקובץ ה README:

```
char a = '5';
char b = '8';

int i = a*b;
printf("%c * %c = %d\n",a,b,i);
```

מדוע קטע הקוד לא ידפיס את מכפלת המספרים?.
הציעו כיצד לתקן את קטע הקוד כך שמכפלת המספרים תודפס. שתי השורות הראשונות יישארו ללא שנוי.
שנו את מינימום הקוד הנדרש.

שאלה 2 (80 נקודות)

בסעיפים הבאים תתרגלו עבודה עם טבלת ה - ASCII, פונקציות קלט ופלט, והמרות בין תווים ומספרים.

2.1 (5 נקודות) PrintAscii.c

כתבו תכנית המדפיסה את טבלת ה ASCII החל מתו 32 ועד תו 127. על הפלט של תכניתכם להראות בדיוק כמו בתכנית ~labc/www/ex1/PrintAscii

שם הקובץ התכנית יהיה PrintAscii.c

2.2 (15 נקודות) CheckParenthesis.c

בתכנית זו יהיה עליכם לבדוק תקינות סוגריים במחרוזת.

כתבו תכנית המקבלת מן המקלדת מחרוזת המסתיימת בתו *. העזרו בפונקצייה getchar() על מנת לקרוא את המחרוזת תו אחרי תו.

תכניתכם תבדוק את תקינות הסוגריים במחרוזת.
במידה והמחרוזת תקינה הדפיסו valid\n וסיימו אחרת הדפיסו invalid\n וסיימו

הניחו את ההנחות הבאות בנוגע למחרוזת אותה תקבלו. לא מוטל עליכם לטפל (ולא נבחן את תכניתכם) במחרוזות שאינן עונות להנחות הללו.

- במחרוזת יופיעו לכל היותר 4 סוגי סוגריים: < >, { }, [ ], ( ). אלו סוגי הסוגריים שייתכנו במחרוזת.
- במחרוזת יופיעו לכל היותר 10 סימני סוגריים (מארבעת הסוגים שמנינו לעיל).
- מלבד התו * המסמן סוף מחרוזת ותווי הסוגריים הניחו כי המחרוזת תכיל אותיות (קטנות או גדולות)
- שרשור סוגריים הינו חוקי. למשל: abc(xyz(mm))vv
- מחרוזת ריקה (המכילה את סימן הכוכבית בלבד) או ללא סוגריים כלל הינה חוקית.

הניחו כי המחרוזת מסתיימת בסימן *, הינכם יכולים להניח כי לאחר תו זה אין תווים נוספים.

לדוגמא

(השורה הראשונה היא הקלט שהתקבל מהמקלדת, ולאחריו הוקש ENTER, השורה השנייה היא הפלט של תכניתכם)

```
> CheckParenthesis
aaa(b)*
valid

> CheckParenthesis
()<>*
valid

> CheckParenthesis
aa(bbb{ccc*
invalid

> CheckParenthesis
aa(bb<cc>dd){ee}*
valid
```

שם הקובץ התכנית יהיה CheckParenthesis.c

2.3 (5 נקודות) ChangeCase.c

כתבו תכנית המקבלת כקלט מן המקלדת מחרוזת, הניחו כי המחרוזת יכולה להכיל תווים כלשהם ללא הגבלה ומסתיימת בנקודה.

על התכנית להפוך כל אות גדולה לקטנה וכל אות קטנה לגדולה כבדוגמא הבאה:

```
>ChangeCase
Exercise #1 labc 2009.
eXERCISE #1 LABC 2009.
```

שם הקובץ התכנית יהיה ChangeCase.c

2.4 (55 נקודות) ChangeBase.c

בשאלה זו יהיה עליכם לממש המרת בסיסים פשוטה.

תכניתכם תקבל כקלט מספר, הבסיס בו הוא מיוצג, והבסיס אליו אתם מעוניינים להמיר את המספר.

פלט התכנית יהיה המספר בבסיס החדש.

המחרוזת אותה תקבלו כקלט תהיה בתבנית הבאה
OriginalBase#NumberInOriginalBase#NewBase#

לדוגמא

10#932#4#
המספר 932 בבסיס 10, הבסיס החדש הוא 4

2#010100#9#
המספר 010100 בבסיס 2, הבסיס החדש הוא 9

הנחות על הקלט:

- הניחו כי OriginalBase ו NewBase הינם בין בטווח 2-10.
- NumberInOriginalBase הינו מספר אי שלילי בעל 5 ספרות לכל היותר.
- הניחו כי NumberInOriginalBase הינו ללא אפסים מובילים. למשל לא תצטרכו לטפל במספרים כגון 0012,0000,01234.
- באם NumberInOriginalBase אינו מתאים להגדרת OriginalBase (ספרות חורגות) הדפיסו invalid\n וסיימו את התכנית

לדוגמא

```
> ChangeBase
2#1#10#
1

>ChangeBase
10#30#8#
36

>ChangeBase
2#1001#9#
10

>ChangeBase
2#2123#10#
invalid
```

הנחיות

הינכם רשאים להעזר בכל אלגוריתם המרת בסיס שתמצאו לנכון מהרשת או מהספרות.
הצעה לפתרון: שקלו להמיר את המספרים לבסיס 10 ולאחר מכן לבסיס הדרוש.

הסבירו בקובץ ה README בקצרה באיזו שיטה השתמשתם להמרת הבסיסים.

שם קובץ התכנית יהיה ChangeBase.c

שאלה 3 (8 נקודות) MySqrt.c

משימתכם בשאלה זו תהיה למצוא את השורש הריבועי של מספר נתון.

קיימים מספר אלגוריתמים למציאת שורש ריבועי, בשאלה זו אנו נתמקד באלגוריתם פשוט אותו יהיה עליכם ליישם ושמו Babylonian Method.

ייתרונו של האלגוריתם הזה הינו בפשטותו והאפשרות לקודדו בקלות יחסית.

לצורך הבנת האלגוריתם תוכלו להעזר בקישור הבא (ברשת קיימים הרבה הסברים ודוגמאות)

http://en.wikipedia.org/wiki/Methods_of_computing_square_roots#Babylonian_method

כאמור, ייצוג השורש הריבועי של מספר אינו בהכרח סופי, לצורך הדוגמא השורש הריבועי של המספר 2 אינו מספר רציונלי. לצורך הבטחת סופיות האלגוריתם אנו נגדיר שני תנאי עצירה לאלגוריתמים אותם ניישם.

לצורך כך נפתח בהקדמה קצרה לגבי השוואת מספרים מטיפוס double בשפת C.

בניגוד למספרים שלמים, ייצוג מספרים שבריים במחשב אינו טריוויאלי. מספרים שבריים יכולים להיות כאמור, לא רציונליים או בעלי רמות דיוק שונות. בנוסף לכך סופיות הייצוג במחשב גורמת לכך שלעיתים תוצאות מעוגלות או מקוצצות.

בשל הסיבות אותן פירטנו לעיל, באפליקציות מדעיות נמנעים לרוב מהשוואת מספרים שבריים על ידי אופרטור השוויון (==) ומשתמשים בשיטות חלופיות. אחת השיטות המקובלת להשוואת שני משתנים מסוג double היא כמפורט להלן:

יהיו a,b שני מספרים מסוג double

נגדיר a==b באם מתקיים |a-b| < epsilon, כאשר epsilon הינו קבוע המוגדר על ידינו. למשל 0.0001

נשוב לאלגוריתם מציאת השורש הריבועי.

1. בהנתן מספר N – אנו רוצים למצוא את שורשו הריבועי.
   נניח כי באיטרציה הנוכחית באלגוריתם המספר S הוא המייצג את השורש הריבועי של N
   אנו נעצור את האלגוריתם (נחשיב את S כשורש של N) באם מתקיים $|S^2 - N| < epsilon$

2. נגדיר max iterations את מספר האיטרציות המקסימלי אותו נבצע. דהיינו האלגוריתם יסיים בכל מקרה לאחר מספר איטרציות זה, גם אם תנאי (1) לא התקיים.

עליכם לממש בקובץ MySqrt.c את הפונקצייה:

```
double sqrtBabylonian(double n, double epsilon, int maxIterations);
```

הפונקצייה תחזיר את השורש הריבועי של n.
הניחו כי הפרמטרים אותם תקבלו הינם תקינים.
במידה ו n הינו מספר שלילי החזירו 1-.

שימו לב כי את הערך ההתחלתי (שלב 1 באלגוריתם), אתם רשאים להעריך כרצונכם.
פרטו בקובץ ה README באיזו שיטה בחרתם להערכה זו.

הקובץ אותו תגישו MySqrt.c יכיל את הפונקצייה, ופונקציית main המדגימה את הקריאה ל-sqrtBabylonian עם פרמטרים כלשהם, לפי שיקולכם ומדפיסה את התוצאה.

אינכם מחוייבים לפלט בפורמט כלשהוא בפונקציית ה main – מטרתו היא להדגים את הקריאה ל sqrtBabylonian.

<u>הגשה</u>

קראו את ההנחיות להגשת התרגילים.

עליכם להגיש את הקבצים הבאים בלבד (בתרגיל זה אין להגיש קבצים נוספים):

README
MyMagicSquare.c
PrintAscii.c
CheckParenthesis.c
ChangeCase.c
ChangeBase.c
MySqrt.c

קבצו את הקבצים יחדיו לקובץ בשם ex1.tar (הוראות תוכלו למצוא בקישור

http://moodle.cs.huji.ac.il/cs09/mod/resource/view.php?id=394

השתמשו בסקריפט ~labc/www/ex1/check_ex1 על מנת לוודא תקינות קובץ ההגשה.

הגישו את הקובץ ex1.tar דרך מערכת ה moodle (יש לבצע login על מנת לראות את לחצן ההגשה)

בהצלחה