תרגלו את שפת ה C שלכם - אסופת תרגילים לפי רמות

טל אלון

רמה 0

תרגיל 0-1) כתבו תכנית שלוקחת מהמשתמש מספר שלם (שלילי או חיובי) ומדפיסה את הערך המוחלט של המספר.

תרגיל 0-2) כתבו תכנית שלוקחת משמשתמש שני מספרים שלמים חיוביים ומדפיסה את תוצאת הכפל של שני המספרים הללו. יש לכתוב את התכנית ללא שימוש בפעולת הכפל הבנויה בשפת C (רמז: יש להשתמש בלולאה).

תרגיל 0-3) כתבו תכנית שלוקחת מהמשתמש תו (char) אחד. אם התו הוא אות קטנה התכנית תדפיס את אותה האות אבל גדולה, במקרה של כל תו אחר התכנית תדפיס את התו ללא שינוי.

תרגיל 0-4) כתבו תכנית שלוקחת תו אחד (צריך להיות אות גדולה), ומדפיסה את כל האותיות הגדולות החל מ A עד לאותה האות.

תרגיל 0-5) כתבו תכנית שלוקחת שני תווים - שתי אותיות גדולות או שתי קטנות - ומדפיסה את כל האותיות שביניהן.

רמה 1

תרגיל 1-1)

Will the following program print the value 2 ?

```
#include <stdio.h>

void my_plus(int x)
{
    x++;
}

void main()
{
    int i;

    i=1;

    my_plus(i);

    printf("%d\n",i);
}
```

will the printed value change if the function 'my_plus' will receive 'int i' as a parameter instead of 'int x' ?

Why is that ?

תרגיל 1-2)

התכנית, בשם fixit.c, אמורה להדפיס את שורת הפלט:

c*c = 25.000000

תקנו את התכנית כך שתבצע את המוטל עליה ...

```
#include <Stdio.h>
integer main ()
{
  float a, b; c;
  a = 3;
  b = 4.0:
  c = a*a + b*b
  print (“c*c = %d/n”, c);
  Return 0;
}
```

<u>הערה</u>: נסו למצוא טעויות בעצמכם, מתוך קריאת התכנית. פנו לעזרת הקומפיילר רק כשלב אחרון ...

<u>תרגיל 1-3)</u>

כתבו תכנית בשם sumnum שמקבלת כקלט מספר תלת-ספרתי (כלומר בעל שלוש ספרות עשרוניות) ומדפיסה את השארית המתקבלת מחלוקת המספר בסכום הספרות שלו. לדוגמה:

קלט: 123

פלט: 3 (כיוון ש: $123 \bmod (1 + 2 + 3) = 3$)

קלט: 864

פלט: 0

בדקו את המספר שאתם מקבלים כקלט. אם הוא איננו תלת-ספרתי, עליכם להדפיס הודעת שגיאה ולסיים את התכנית. למשל:

קלט: 67

פלט: !Invalid input. Should be a 3-digit number

<u>תרגיל 1-4)</u>

כתבו תכנית בשם convert_cm, שקוראת מספר ממשי חיובי (המייצג אורך), ולאחריו – מופרד ע"י רווח – אחד מן התווים הבאים: m, f, i או y, המייצג את יחידת המידה (meter, foot, inch או yard). התכנית מחשבת ומדפיסה את האורך בסנטימטרים, כאשר היא מבצעת את ההמרה על-פי הטבלה הבאה:

1 meter = 100 cm

1 foot = 30.28 cm

1 inch = 2.54 cm

1 yard = 91.44 cm

לדוגמה (שימו לב לרווח המפריד):

קלט: 20.4 i

פלט: 51.816 cm

מלבד המקום בו אתם בודקים שהאורך חיובי, יש לכתוב את התכנית ללא שימוש ב- if-else !

<u>הערה</u>: הקפידו לבדוק את הקלט: כזכור, אסור שהאורך יהיה מספר שלילי וכן על יחידת-המידה להיות חוקית. אם לא, הדפיסו הודעת שגיאה וסיימו את התכנית.

<u>תרגיל 5-1)</u>

כתבו תכנית בשם stats שקוראת רצף של מספרים ממשיים אי-שליליים, עד שהיא נתקלת במספר שלילי – במקרה זה היא עוצרת, ומדפיסה את הסטטיסטיקות הבאות על המספרים שקראה (לא כולל המספר האחרון, השלילי): המספר המקסימאלי, המספר המינימאלי, הסכום של כל המספרים, הממוצע של המספרים.

רמה 2

תרגיל 2-1)

Write a program that reads a list of distinct numbers and outputs the five largest numbers in the list. The input is a list of n numbers separated by spaces, a_1 a_2 ... a_n. The program should output the five largest numbers, printed from largest to smallest (with two digits precision), separated by spaces. Your C code file name should be max5.c
Example:

input:

110 -0.55 1.19 200 -7 456.76 2.12

output:

456.76 200 110 2.12 1.19

תרגיל 2-2)

Write a program that reads a sequence of positive integers and outputs the length of the longest strictly increasing sub-sequence of consecutive numbers. The input is a sequence of integers given in one line, separated by spaces. The output is a number indicating the length of the longest strictly increasing sub-sequence of consecutive numbers contained in the given sequence. Your C code file name should be sequence.c
Example:

input:

23 23 2 56 456 500 8 90 81 500 34 100 93 23445 1

output:

4

תרגיל 3-2)

Write a program that reads text characters from the input until an EOF occurs. The program should then output the number of times each letter appeared in the input, in a tabular form. Capital letters and lower case letters should be counted together. The output should consist of 26 lines. Each line starts with a lower case character, then a colon, a space, and the accumulated times the letter (both its lower case and its upper case) has appeared in the input. Your C code file name should be count.c

Example:

input:

Abracadabra

output:

a: 5
b: 2
c: 1
d: 1
e: 0
...
q: 0
r: 2
s: 0
...
z: 0

תרגיל 4-2)

Write a program that reads an integer number and outputs one digit. The input is an integer n_0 (n_0 may be either positive or negative). The program calculates a series of integers n_1, n_2, ... such that n_{i+1}=(sum of digits of n_i). The last element of the series should be a single digit d. The program should output the digit d. Your C code file name should be reduce.c .
Example:

input:

5789

(internal calculations - should not be printed: n_0=5789, n_1=29, n_2=11, n_3=2, d=n_3=2, stop)

output:

2

<u>תרגיל 2-5)</u>

כתבו תכנית בשם palindrome.c, שקוראת מחרוזת קלט (משפט שלם, כולל רווחים המסתיים בלחיצת Enter של המשתמש), ומחליטה האם המחרוזת היא פלינדרום או לא. להזכירכם – פלינדרום הוא משפט שניתן לקרוא אותו מן ההתחלה לסוף או מן הסוף להתחלה מבלי שישתנה. עליכם להיות case-insensitive, כלומר לא להבחין בין אותיות לטיניות גדולות וקטנות, וכן עליכם להתעלם מרווחים ומכל סימן אחר שאיננו אות. לדוגמה:

קלט: ABBA

פלט: This is a palindrome!

קלט: ABCA

פלט: This is NOT a palindrome.

קלט: Madam, I'm Adam

פלט: This is a palindrome!

קלט: Dennis and Edna sinned.

פלט: This is a palindrome!

<u>הערה</u>: הגבילו את אורך מחרוזת הקלט ל-200 תווים, ובדקו שאינכם קוראים יותר תווים מכך.

<u>תרגיל 2-6)</u>

כתבו פונקציה רקורסיבית בעלת המפרט (prototype) הבא:

void factorize (int n);

הפונקציה מקבלת מספר שלם וחיובי *n*, ועוברת על כל המחלקים הפוטנציאליים שלו (על כל ערכי *k* מ- 2 ועד לערך השלם של שורש n, שימו לב שאין צורך לבדוק עד 1 – *n* . ברגע שהיא מוצאת מספר *k* שמחלק את *n* ללא שארית, היא מדפיסה אותו, ואז ממשיכה לפרק ברקורסיה את . אם הפונקציה לא מוצאת אף מחלק כזה, סימן ש- *n* ראשוני, ואז היא פשוט מדפיסה אותו ומסיימת.

תוך שימוש בפונקציה שכתבתם, כתבו תכנית בשם factorize.c, שקוראת מספר חיובי כקלט, ומדפיסה את הפירוק שלו לגורמים ראשוניים. לדוגמה:

קלט: 561

פלט: 3 11 17 (בדקו ותראו כי 561 = 3*11*17)

קלט: 700

פלט: 2 2 5 5 7 (שימו לב שהגורמים 2 ו-5 מופיעים פעמיים)

קלט: 31

פלט: 31 (כי 31 הוא מספר ראשוני)

<u>תרגיל 2-7)</u>

כתבו פונקציה רקורסיבית בעלת המפרט (prototype) הבא:

```
int fib_elem (int a0, int a1, int n);
```

על הפונקציה להחזיר את האיבר ה-n (כאשר n מספר אי-שלילי כמובן) בסדרת פיבונאצ'י ששני איבריה הראשונים נתונים על-ידי a_0 ו-a_1 . להזכירכם, איבר זה הוא סכום שני האיברים הקודמים, כלומר עבור כל $n > 1$ מתקיים:

$$a_n = a_{n-1} + a_{n-2}$$

כתבו תכנית בשם gen_fib.c שקוראת מן המשתמש את ערכי a_0, a_1 ו-n (יש לבדוק ש-n איננו שלילי), ומדפיסה את האיבר ה-n (זכרו שהאיבר מס' 0 הוא האיבר הראשון) בסדרת פיבונאצ'י ששני הם a_0 ו-a_1 . לדוגמה:

קלט: 6 1 0

פלט: 8(כי הסדרה היא: … 0, 1, 1, 2, 3, 5, 8,)

קלט: 8 2 1-

פלט: 29(כי הסדרה היא: … 1-, 2, 1, 3, 4, 7, 11, 18, 29)

<u>תרגיל 2-8)</u>

כתבו פונקציה בעלת המפרט הבא:

```
void mat_multiply (double A[DIM][DIM], double B[DIM][DIM], double C[DIM][DIM]);
```

הפונקציה מבצעת כפל של המטריצה A במטריצה B, כאשר שתי המטריצות הן מטריצות ריבועיות בגודל DIM*DIM, כאשר DIM הוא קבוע אותו תגדירו להיות 3 על-ידי פקודת #define, ושומרת את התוצאה במטריצה הריבועית C. להזכירכם, פעולת כפל-המטריצות מתבצעת על-ידי (לכל $0 \le i, j < DIM$):

$$C[i][j] = \sum_{k=0}^{DIM-1} A[i][k] \cdot B[k][j]$$

כתבו פונקציה נוספת:

```
void mat_print (double A[DIM][DIM]);
```

הפונקציה מדפיסה את כל איברי המטריצה A בצורה נאה (כל שורה של המטריצה בשורת פלט נפרדת).

כתבו תכנית בשם mat_mul.c שמחשבת את הכפל (את המספרים אינכם צריכים לקרוא כקלט מן המשתמש, פשוט הכניסו אותם למטריצות בתוך ה-main):

$$\begin{pmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0.6 & 0 & 0.8 \\ 0 & 1 & 0 \\ -0.8 & 0 & 0.6 \end{pmatrix}$$

ואז מדפיסה את המטריצה שהתקבלה כפלט.

<u>תרגיל 9-2)</u>

כתבו תוכנית המחשבת את סכום הריבועים של סדרת מספרים שלמים אי שליליים.

<u>קלט</u>: סדרת מספרים שלמים אי שליליים (מספר שלילי יסמן את סוף הקלט).

<u>פלט</u>: סכום הריבועים של סדרת המספרים.

<u>דוגמא</u>:

<u>קלט</u>: 19- 3 7 6 10

<u>פלט</u>: 194

תרגיל 10-2)

כתבו תוכנית המחשבת עבור סדרת מספרים את מספר המספרים בסדרה הקטנים ממש מהמספר האחרון בסדרה.

<u>קלט</u>: מספר טבעי n הקטן מ-1234, וסדרה של n מספרים שלמים.

<u>פלט</u>: מספר המספרים בסדרה הקטנים ממש מהמספר האחרון בסדרה.

אם n אינו בתחום שהוגדר, התוכנית תדפיס הודעת שגיאה.

<u>דוגמא</u>:

<u>קלט</u>: 13 3 29 1 25 11 7 7 27 4 17 11

<u>פלט</u>: 6

<u>תרגיל 11-2)</u>

כתבו תוכנית המחשבת את כל המחלקים של מספר טבעי נתון ובודקת האם הוא ראשוני או פריק.

<u>קלט</u>: מספר טבעי.

<u>פלט</u>: רשימה ממוינת של כל המחלקים של המספר ואינדיקציה האם הוא ראשוני או פריק.

<u>דוגמאות</u>:

<u>קלט</u>: 12

<u>פלט</u>:

1

2

3

4

6

12

12 is a composite number

<u>קלט</u>: 13

<u>פלט</u>:

1

13

13 is a prime number

תרגיל 2-12)

כתבו תוכנית המתרגמת מספר לייצוג בינארי.

קלט: מספר טבעי.

פלט: הייצוג הבינארי של המספר, ומספר הספרות הבינאריות.

דוגמא:

קלט: 309

פלט:

100110101

The number of binary digits is: 9

תרגיל 2-13)

כתבו תוכנית הקוראת מספר שלם אי שלילי n, ומדפיסה את כל זוגות המספרים החברים הקטנים מ-n.

סיבוכיות זמן ריצת התוכנית - $O(n^{3/2})$.

דוגמא:

קלט: 3000

פלט:

220 , 284

1184 , 1210

2620 , 2924

תרגיל 2-14)

מימוש נפת אריסטוסטנס: כתבו תוכנית הקוראת מספר שלם אי שלילי n, ומדפיסה את כל המספרים הראשוניים הקטנים או שווים ל-n. ניתן להניח כי $n \leq 10^6$.

מה סיבוכיות זמן הריצה?

דוגמא:

קלט: 9

פלט:

2

3

5

7

There are 4 prime numbers lower or equal to 9

תרגיל 15-2)

כתבו שתי פונקציות הפועלות על מחרוזות כדלהלן:

int my_strcmp(char *s,char *t);

int my_strstr(char *s,char *t);

הפונקציה my_strcmp מחזירה 1- אם המחרוזת s קטנה לקסיקוגרפית מהמחרוזת t , אפס אם הן שוות ו 1 אם s גדולה לקסיקוגרפית מ-t.

הפונקציה my_strstr תחזיר 1 במידה והמחרוזת t מופיעה כתת מחרוזת ב-s. אחרת, הפונקציה תחזיר 0.

דוגמאות:

my_strcmp("abc","abd") תחזיר 1-

my_strcmp("abc","abcd") תחזיר 1-

my_strcmp("Abc","ABC") תחזיר 1

my_strstr("abcxyz","cxy") תחזיר 1

my_strstr("abcxyz","cxyy") תחזיר 0

יש לשלב את הפונקציות בתוכנית ראשית הקוראת שתי מחרוזות ומדפיסה את תוצאות הרצן של שתי הפונקציות.

הערה: במימוש my_strcmp ו-my_strstr אין להשתמש בפונקציות ספריה.

<u>תרגיל 16-2)</u>

כתבו פונקציה המתרגמת מחרוזת המכילה מספר ממשי לערך המספרי של אותו מספר.

```
double string_to_real(char s[]);
```

לדוגמה, string_to_real("123.45") תחזיר 123.45

<u>הערה</u>: במימוש string_to_real אין להשתמש בפונקציות ספריה.

רמה 3

תרגיל 3-1)

Write a program that reads a sequence of integers and outputs the length of the longest strictly increasing sub-sequence. The input is a sequence of up to 100 integers given in one line, separated by spaces. The output is a number indicating the length of the longest strictly increasing sub-sequence (not necessarily consecutive) contained in the given sequence.
Your C code file name should be adseq.c

Example:
input:
340 23 2 56 456 -912 500 8 90 81 400

output:
4

תרגיל 3-2)

Write a desk calculator program. It should read an expression in algebraic notation, and print as output the polish/prefix notation representation of that expression and evaluate it. Your calculator should handle only integer operands (but use float or double as its internal representation), and should support the binary operations +, *, -, and /. You are encouraged to use a recursion implementation.

The input should be a fully parenthesized expression. Spaces are allowed and should be ignored. Your program should read text characters from the standard input until an EOF occurs and store the input expression in an array of characters (maximum length of expression may be assumed to be 100).
Your output should include two separate lines: The first line should include the prefix notation representation of the input expression. There should be a space following each operator or operand. No parentheses should appear in the output. The second output line should be the value of the input expression, printed as float with 3 digit precision.
Your C code file name should be calculator.c

Example:
input:
(((3+4) - (5+6)) * 5)

output:
* - + 3 4 + 5 6 5
-20.000

תרגיל 3-3)

בתרגיל זה עליכם לפתור בעיה בגאומטריה. נגדיר את המבנים הבאים:

```
typedef struct
{
   int X, Y;
} Point;

typedef struct
{
   Point LeftPoint, RightPoint;
} Line;
```

במבנה האחרון תמיד מתקיים: RightPoint.X => LeftPoint.X.
כתבו תוכנית FindIntersections :
קלט: קבוצת קטעים במישור, כל קטע (Line) מיוצג ע"י 4 מספרים ממשיים מטיפוס double. בקלט כל שורה (בת ארבעה איברים) תייצג קטע בפורמט : left_x, left_y, right_x, right_y)).
פלט: מספר המפגשים (הצטלבויות) בין קטעים שונים. אם שני קטעים זהים או חופפים, יחשב הדבר למפגש יחיד.
השתמשו ברשימה מקושרת חד כיוונית כמבנה הנתונים הראשי לשמירת הקטעים.

תרגיל 3-4)

בשני תרגילים אלו הנכם מתבקשים לחקור את שכיחות המילים השונות בטקסט אנגלי פשוט ולהציג את ההתפלגות בצורה גראפית. נגדיר

```
typedef struct
{
   char * mot; // word
   unsigned F; // frequency
} Word;

typedef struct wordNode
{
   Word data;
   WordNode *next;
} WordNode;
```

כתבו שתי תוכניות (שמותיהן להלן) אשר יבצעו את הדברים הבאים (כל אחת מהן):

- תקרא קבץ טקסט,
- "תקלף" מהמילים שבו את סימני הפיסוק, (ראה תרגיל קודם)
- תהפוך את כל האותיות ל lower case
- תוסיף את המילים
  - o לרשימה מקושרת חד כיוונית (תרגיל 2)- תוכנית בשם UseList.
  - o לעץ חיפוש בינארי (תרגיל 3)- תוכנית בשם UseTree.
- תדפיס בסדר מילוני את כל המילים כאשר עבור כל מילה תודפס המילה, מספר מופעיה (בטור מיושר לימין), ומימין מספר כוכביות כמספר מופעיה.

לדוגמה:

```
a                              2**
antidisestablishmentarionism   1*
if                             1*
is                             4****
of                             1*
```

```
music                          2**
symphony                        1*
the                           5*****
this                          2**
ulichshetarnegoloteihem         1*
```

אם מספר המופעים הוא כזה שיגרום גלישת כוכביות לשורה הבאה, יש להשתמש באות T כמייצגת 10 כוכביות.
תכנון הקוד צריך להיות מודולרי, התכנית הראשית חייבת להיות קצרה (כמה הגדרות משתנים וכמה קריאות לפונקציות).

<u>תרגיל 5-3)</u>

יש לכתוב תוכנית בשם indent שתקבל קובץ c ותעשה לו אינדנטציה כמקובל. שורת הפקודה תהיה indent file1 file2 כאשר file1 הינו קובץ c, והפלט ייכתב לקובץ file2.
כל רמת אינדנטציה תתבטא ב-tab אחד.
אינדנטציה נדרשת עבור קוד בתוך בלוקי – if, else, for, while, switch, do וכו' וכל בלוק חדש. שימו לב שיתכן כמובן קינון.