מכללה אקדמית הדסה
החוג למדעי המחשב

תרגיל 7# בקורס תכנות מודולרי א'
משתנים תוויים ומחרוזות

תכנית 1#: התאמה חלקית של מחרוזות (30%)

בתכנית זאת עליכם לממש את הפונקציה inexact_pos, ולכתוב תכנית ראשית אשר כל תפקידה בחיים הוא לוודא שהפונקציה הנ"ל פועלת כהלכה. אנו אומרים שהתכנית הראשית משמשת driver אשר מפעיל את הפונקציה (כמו שהנהג באוטו מפעיל את המכונית).

פרוטוטיפ הפונקציה inexact_pos הוא כדלהלן:

```
int inexact_pos(const char pattern[],
                const char text[],
                int from_pos,
                int mismatch)
```

הפונקציה מחפשת את התבנית (הסטרינג) שהועבר לה בפרמטר הראשון (pattern) בתוך הסטרינג שהועבר לה בפרמטר השני (text), החל מהמקום מספר from_pos. ההתאמה לא בהכרח חייבת להיות מושלמת: הפרמטר הרביעי mismatch מורה מה איכות ההתאמה שצריכה להימצא - כמה תווים בטקסט מותר שיהיו שונים מהתבנית, ועדיין הדבר ייחשב שיש התאמה. על הפונקציה להחזיר את מספרו של התא הראשון בטקסט אשר בו מתחיל קטע של הטקסט המתאים לתבנית, או 1- אם לא נמצאה התאמה. במידה וקיימות מספר התאמות יש להחזיר את הראשונה ביניהן.

<u>דוגמות</u>:

inexact_pos("abc", "abdabc", 0, 0) should return 3

שכן: יש להתחיל לחפש את abc החל מהמקום 0# ב- abdabc אך נדרשת התאמה מושלמת, וכזאת נמצאת החל מהתא 3#.

inexact_pos("abc", "abdabc", 0, 1) should return 0

שכן: יש להתחיל לחפש את abc החל מהמקום 0# ב- abdabc אך עתה מותר שלא תהא התאמה מושלמת: גם אם יש הבדל בתו יחיד עדיין זו נחשבת התאמה. ולכן התאמה רצויה נמצאת כבר החל מהתא 0#.

inexact_pos("abc", "abdabxcab", 0, 0) should return -1

שכן: עתה אנו דורשים התאמה מוחלטת, וכזאת לא נמצאת בכל הטקסט.

inexact_pos("abc", "abcabc", 1, 0) should return 3

שכן: עתה אנו שוב דורשים התאמה מושלמת, אך מחפשים בטקסט החל מהתא 1#, ולכן תת-טקסט כנדרש נמצא רק החל בתא 3#. (אגב, אם היינו מבקשים לחפש החל בתא 1#, ומאפשרים חוסר התאמה במידה 3, אזי היה מוחזר הערך 1).

במידה והטקסט קצר מהתבנית יוחזר הערך 1-.

התכנית הראשית שתכתבו תקבל באמצעות argv ארבעה פרמטרים:

א. התבנית הרצויה.

ב. הטקסט הרצוי.

ג. המיקום בטקסט בו יש להתחיל לחפש את התבנית, (יתקבל כסטרינג [כמו כל איבר ב- []argv), ויומר על-ידי התכנית למספר טבעי).
ד. מידת חוסר ההתאמה המותרת (כמו קודמו).

התכנית תדפיס כפלט את הערך המוחזר על-ידי הפונקציה (וכרגיל, תעבור שורה).

<u>הערות:</u>

א. יש לוודא שמספר הפרמטרים שהתכנית מקבלת באמצעות argv הוא כנדרש. במידה והוא אינו כנדרש יש להוציא את הודעת השגיאה:
```
cerr << "Usage: ex7a <pattern> <text> <from pos> <mismatch>" <<endl ;
```
ולעצור את ביצוע התכנית.
ב. יש לוודא כי זוג הפרמטרים האחרונים, אכן מבטא מספרים, ומספרים טבעיים. במידה והם אינם כנדרש יש לתת הודעת שגיאה כמו קודם, ולעצור את ביצוע התכנית (רמז: ניתן לעשות שימוש בטבלת הascii).
ג. כל הרצה של תכנית תקבל ארבעה ערכים כמתואר, תבדוק אותם, תזמן את הפונקציה, תציג פלט ותעצור.

<u>תכנית 2#: טיפול במספרים טבעיים גדולים (55%)</u>

בתכנית זאת עליכם לממש מחשבון עבור מספרים **טבעיים** גדולים מאוד, (למשל כאלה בני עשרות ספרות). כדי לעשות זאת נייצג כל מספר באמצעות מערך של תווים; כל תא במערך יכיל ספרה של המספר. לדוגמה: המספר 576 ייוצג באמצעות המערך הבא:

0	5	7	6
#3	#2	#1	#0

כלומר ספרת האחדות תשמר בתא 0#, ספרת העשרות בתא 1#, וכן הלאה. בתאים בהם אין שימוש יישמר הערך אפס. (שימו לב כי ציירנו כאן את תאי המערך ההפך מדרך הרגילה: התא 0# צויר מימין, במקום משמאל, כמו בד"כ).

מבנה הנתונים המרכזי של התכנית יהיה לפיכך מערך דו-ממדי של תווים בגודל:
```
const MAX_NUMBERS = 10,
      MAX_DIGITS = 20 ;
```

הפעולות שהתכנית תאפשר למשתמש:

א. הזנת מספר נוסף לרשימת הנתונים שהתכנית מחזיקה. המשתמש יזין את האות i כדי לציין שזו הפעולה הרצויה (insert), ואחר את המספר שברצונו להזין. דוגמא(ות):

i 1345843444
i 234
i 4352345

המספר ייקרא כסטרינג, אין לקרוא ערך לתוך משתנים מספריים. אתם רשאים להחזיק מערך עזר בן MAX_DIGITS +1 תאים. אחרי קריאת הערך לסטרינג יש להפכו (reverse) לתוך מבנה הנתונים המרכזי של התכנית, ושם לשמרו ללא ה- '\0' (שכן התכנית אינה עוסקת במחרוזות, אלא במערכים של תווים). המספרים יאוחסנו לפי הסדר: מספר חדש יקלט כל פעם לשורה הבאה.

ב. חיבור שני מספרים שהוזנו, והצגת סכומם. המשתמש יזין את התו + כדי לבקש פעולה זאת, ואחר את האינדקסים (במערך המספרים) של שני המספרים שברצונו לחבר. דוגמא (המשך לדוגמא מהסעיף הקודם)

+ 2 0

פירושו: חבר את המספר הנמצא בשורה מס' 2 עם המספר הנמצא בשורה מס' 0.

התכנית תציג את שני האופרנדים, ואת סכומם. האופרנדים יוצגו בסדר בו הם הוזנו על-ידי המשתמש, ואחר יוצג הסכום. הפלט יופיע בפורמט:

<first operand> + <second operand> = <sum>

אם נמשיך את הדוגמאות שנתנו עד כה, יודפס:

4352345 + 1345843444 = 1350195789

במידה והסכום גדול מכדי להיות מוחזק במערך בן MAX_DIGITS תאים, יוצגו בכל אופן הספרות הנמוכות של התוצאה – אותן אין לנו בעיה לחשב, ולאחר מכן רווח והאות האנגלית הקטנה o. זאת כדי להתריע שחלה גלישה (overflow).

ניתן להניח כי הקלט תקין, כלומר מוזנים רק מספרי שורות המכילים מספרים שכבר הוזנו. שימו לב, גלישה **בתוצאה** של הפעולה, אינה הופכת את הקלט לבלתי תקין מבחינתנו.

ג. חיסור שני מספרים שהוזנו, והצגת הפרשם. הקלט הוא כמו קודם בחיבור, כאשר התו "–" מסמן את הפעולה הרצויה. יש לחסר את המספר השני מהראשון. התוצאה תוצג בפורמט כמו שתואר עבור חיבור. במידה והמחוסר קטן מהמחסר יש להציג את התוצאה כאילו בוצע חיסור רגיל, בתוספת הסימן u (לציון underflow) לצד ההפרש.

הסבר נוסף על overlfow ו underflow ינתן בתרגול.

ד. הצגת המספרים שהוזנו. כל מספר יוצג בשורה נפרדת. קוד הפעולה הוא התו l (בשביל list).

ה. סיום. קוד הפעולה e.

<u>הערות:</u>

ביצוע הפעולות החשבוניות יהיה על פי עקרונות של חיבור/חיסור ארוך.

במהלך כל התכנית הניחו תקינות קלט מלאה, מלבד יוצא דופן אחד: overflow & underflow שלגביהם פורט לעיל.

בכל המקומות בהם אתם מציגים מספרים, אין להציג אפסים מובילים. מלבד יוצא דופן אחד: המספר 0 עבורו תוצג המחרוזת "0" ולא מחרוזת ריקה.

אין להציג כל תפריט שהוא למשתמש.

התכנית הראשית לא תכלול פקודות פלט. כל הפלטים יוצגו באמצעות פונקציות.

הקפידו על חלוקה נבונה לפונקציות, ועל שימוש מושכל בנושאים אחרים שנלמדו בעבר (לולאות, enum, העברת משתנים לפוקנציות).

הכרת מערכת ההפעלה לינוקס – 15%

1. מה עושה הפקודה : ls -l /cs/stud
   מה עושה הפקודה more
   מה עושה הפקודה : ls -l /cs/stud | more
   מהו, באופן כללי : pipe
2. ברצונכם לדעת היכן במערכת הקבצים במחשב מצוי הקובץ iostream, או כל קובץ אחר. כיצד תבדקו זאת.

נוהל ההגשה :

א. כמקובל.

ב. אם מתכנתים לא מעתיקים. בשביל זה אין חברים!