סדנת תכנות בשפת **C** ,מס׳ קורס 67316 -2018

תרגיל 2

תאריך הגשה: יום חמישי 15.11.18 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): יום שישי 16.11.18 עד שעה 23:55

תאריך ההגשה של הבוחן: יום חמישי 15.11.18 עד שעה 23:55

הנחיות חשובות לכלל התרגילים:

- בכל התרגילים יש לעמוד בהנחיות הגשת התרגילים וסגנון כתיבת הקוד. שני המסמכים נמצאים באתר הקורס – הניקוד יכלול גם עמידה בדרישות אלו.
- בכל התרגילים עליכם לכתוב קוד ברור. בכל מקרה בו הקוד שלכם אינו ברור מספיק עליכם להוסיף הערות הסבר בגוף הקוד. יש להקפיד על תיעוד (documentation) הקוד ובפרט תיעוד של כל פונקציה.
- במידה ואתם משתמשים בעיצוב מיוחד או משהו לא שגרתי, עליכם להוסיף הערות בקוד המסבירות את העיצוב שלכם ומדוע בחרתם בו.
- עבור כל פונקציה בה אתם משתמשים, עליכם לוודא שאתם מבינים היטב מה הפונקציה עושה גם במקרי קצה (התייחסו לכך בתיעוד). ובפרט עליכם לוודא שהפונקציה הצליחה.
- בכל התרגילים במידה ויש לכם הארכה, או שאתם מגישים באיחור. חל איסור להגיש קובץ כלשהוא בלינק הרגיל (גם אם לינק **overdue** טרם נפתח) .מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.
- אין להגיש קבצים נוספים על אלו שתדרשו .ובפרט אין להגיש קובץ README אלא אם צוין במפורש שיש צורך בכך (לדוגמא, בתרגיל זה אין צורך להגיש).
- עליכם לקמפל עם הדגלים 99Wall -Wextra -Wvla -std=c ולוודא שהתוכנית מתקמפלת ללא אזהרות, תכנית שמתקמפלת עם אזהרות תגרור הורדה משמעותית בציון התרגיל. למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex1.c יש להריץ את הפקודה:

```
gcc -Wextra -Wall -Wvla -std=c99 -lm ex2.c -o ex2
```

- עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי 64-bit (מחשבי האקווריום, לוי, השרת river). חובה להריץ את התרגיל במחשבי בית הספר לפני ההגשה. (ניתן לוודא שהמחשב עליו אתם עובדים הנו בתצורת 64-bit באמצעות הפקודה "uname -a" ווידוא כי הארכיטקטורה היא 64, למשל אם כתוב 86x_64)
- לאחר ההגשה, בדקו את הפלט המתקבל בקובץ ה-PDF שנוצר מהpresubmission script בזמן ההגשה. באם ישנן שגיאות, תקנו אותן על מנת שלא לאבד נקודות.
שימו לב ! תרגיל שלא יעבור את ה **presubmission script** ציונו ירד משמעותית (הציון יתחיל מ-50, ויוכל לרדת) ולא יהיה ניתן לערער על כך.
- בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. בדקו מקרי קצה.
במידה וסיפקנו לכם קבצי בדיקה לדוגמא, השימוש בהם יהיה על אחריותכם. במהלך הבדיקה הקוד שלכם ייבדק מול קלטים נוספים לשם מתן הציון.

השוואת מחרוזות (=רצפים)

בתרגיל זה נעסוק בבעיה של השוואת מחרוזות. בהינתן שתי מחרוזות

$X=x_1x_2...x_n$ $Y=y_1y_2...y_m$

אנחנו מחפשים התאמה בין הרצפים שמביאה למקסימום את מספר האותיות הזהות בשני הרצפים עם מינימום אי התאמות. התאמה היא בעצם עימוד של שני הרצפים. לדוגמה:

```
COMPUTER
||| ||||
COMMUTER
```

גודל ההתאמה הזו הוא 8, כאשר יש 7 אותיות מתאימות (match) ואות אחת שלא מתאימה (mismatch).
בנוסף ניתן להכניס רווח (gap) לרצף לצורך התאמה טובה יותר:

```
COIN
|  |
CA-N
```

שימו לב שבהינתן 2 מחרוזות יש כמה אפשרויות לעימוד שלהן. לדוגמה: AGGCTAGTT ו- AGCGAAGTTT
הנה כמה התאמות אפשריות:

```
AGGCTAGTT- 6 matches, 3 mismatches, 1 gap
||   ||||
AGCGAAGTTT

AGGCTA-GTT- 7 matches, 1 mismatch, 3 gaps
|| | | |||
AG-CGAAGTTT

AGGC-TA-GTT- 7 matches, 0 mismatches, 5 gaps
|| |  | |||
AG-CG-AAGTTT
```

השוואת מחרוזות משמשת אנליזה של רצפים ביולוגיים של DNA, RNA וחלבונים, כאשר בהתאמה טובה יש כמה שיותר התאמות וכמה שפחות אי התאמות ורווחים. ציון ההתאמה בין שני הרצפים נקבע לפי המשקל של match, mismatch, ו- gap.

בתרגיל זה נקרא קובץ עם רצפים ונשתמש באלגוריתם תכנות דינמי כדי לחשב את ההתאמה בין כל זוג רצפים.

שלב 1: קריאת קובץ רצפים
רצפים ניתנים בקובץ טקסט. כל רצף מכיל שורת header שמתחילה ב "<" ואחריה שורה אחת או יותר של אותיות. לדוגמה:

```
> This is sequence 1
COMPUTER

> This is sequence 2
COMMUTER

> This is a long sequence 3
ATCGXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
ATCGXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
ATCGXYZABCDEFG
```

בקובץ זה הרצף האחרון "שבור" לשלוש שורות. עליכם להפוך אותו למחרוזת אחת רצופה ללא '\n'
עליכם להשתמש בהקצאת זיכרון דינמית כדי לקרוא כל רצף מהקובץ למערך של chars.

- ניתן להניח שמספר תווים בשורה לא עולה על 100 וגם מספר רצפים בקובץ לא עולה על 100
- ניתן להניח שהקובץ בפורמט שמתואר ואין צורך לבדוק את נכונותו
- לא ניתן להניח דבר על אורך המחרוזת
- מומלץ להשתמש בפונקציות malloc, realloc

שלב 2: מימוש אלגוריתם תכנות דינמי להשוואת זוג מחרוזות
קלט: 2 מחרוזות $X=x_1x_2...x_n$ $Y=y_1y_2...y_m$
כמו כן נתונים משקלים עבור התאמה (match, **m**), אי התאמה (mismatch, **s**) ורווח (gap, **g**). בסדר הזה,
כלומר התוכנית מקבלת 4 ארגומנטים:

```
CompareSequences <path_to_sequences_file> <m> <s> <g>
```

נניח גם שציון ההתאמה האופטימלית בין $y_1y_2...y_j$ ל $x_1x_2...x_i$ הוא $F(i,j)=S$.
יש 3 אפשרויות להתאים $y_1y_2...y_j$ ל $x_1x_2...x_i$:

1. להתאים $y_1y_2...y_{j-1}$ ל $x_1x_2...x_{i-1}$
ולהוסיף התאמה (match or mismatch) בין x_i ו- y_j

$F(i,j) = F(i-1,j-1) + m$ if $x_i = y_j$
$F(i,j) = F(i-1,j-1) + s$ if $x_i != y_j$

2. להתאים $y_1y_2...y_{j-1}$ ל $x_1x_2...x_i$
ולהתאים בין רווח (gap) ו- y_j

$F(i,j) = F(i,j-1) + g$

3. . להתאים $y_1y_2...y_j$ ל $x_1x_2...x_{i-1}$
ולהתאים בין רווח (gap) ו- x_i

$F(i, j) = F(i-1,j) + g$

אנחנו נבחר באפשרות שנותנת את הערך המקסימלי עבור $F(i, j)$.
לכן נחשב את $F(i, j)$ בטבלה עבור כל הערכים ובסוף נבחר את $F(m, n)$. כדי לשחזר את ההתאמה נשמור בכל תא מצביע לתא שנתן את הציון הטוב ביותר.

אתחול `F(0,0) = 0` :
השורה והעמודה הראשונה בטבלה מאותחלים לפי המשקל של gap:

```
F(i,0) = g*i
F(0,j) = g*j
```

לכל זוג רצפים בקובץ קלט חשבו את ההתאמה הטובה ביותר ותדפיסו את הציון של ההתאמה הטובה ביותר עבור כל זוג. אם יש N רצפים בקובץ עליכם לעשות N*(N-1)/2 השוואות, כלומר כל רצף מותאם עם כל הרצפים האחרים.

Input file:

```
> seq1
GCATGCU
> seq2
GATTACA
```

→ CompareSequences input.txt 1 -1 -1
Score for alignment of sequence seq1 to sequence seq2 is 0

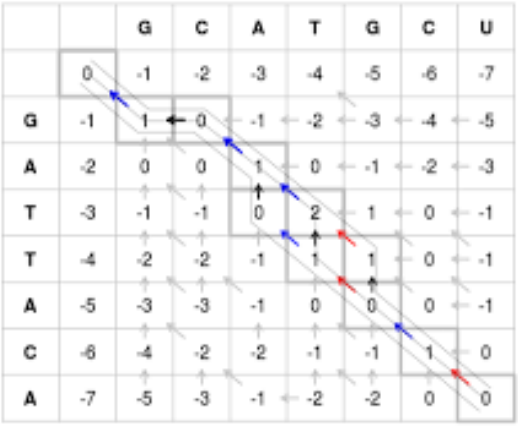
הטבלה המאותחלת תיראה כך:

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1							
A	-2							
T	-3							
T	-4							
A	-5							
C	-6							
A	-7							

אחרי חישוב התכנות הדינמי הטבלה המלאה היא:

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	**0** ← **final score**

שחזור התאמה (בונוס 5 נק׳): כדי לשחזר את ההתאמה צריך לשמור בכל תא בטבלה מצביע לתא שנתן את הציון הטוב ביותר. אם יותר מתא אחד הוביל לאותו ציון, מספיק לשמור מצביע אחד בלבד. לדוגמה:



במקרה זה נדפיס את ההתאמה הבאה (יש עוד אפשרויות):

```
GCA-TGCU
| | | |
G-ATTACA
```

כל התהליך מוסבר כאן: https://www.youtube.com/watch?v=8aJbIh5kKOM

<u>מידע נוסף (כללי)</u>

- חל איסור להשתמש במערכים בגודל דינמי (VLA).
- אתם רשאים להשתמש בכל הספריות הסטנדרטיות של C.
- אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.
- שימו לב שאתם מכירים כל פונקציה בה אתם משתמשים ושאתם בודקים עבור כל פונקציה שהיא הצליחה.
- עליכם לוודא שהקוד שלכם רץ באופן תקין וללא דליפות זכרון. לשם כך עליכם להשתמש בתוכנת **valgrind** (מדריך במודל)

1. <u>טיפול בשגיאות:</u>

- הדפסות שגיאה יודפסו אל הstderr, כמו כן נדרש להבדיל בין שגיאה בתוכנה לבין קלט לא תקין וכדומה. קובץ לא קיים למשל היינה שגיאה בתוכנה שיש לטפל בה ולהחזיר הודעת שגיאה.
- התמודדות עם שגיאות קלט לא תקין נמצאות בתיאור התרגיל, עבור שגיאות שאינן נמצאות בתיאור התרגיל ניתן לפנות לפתרון בית הספר.
- שימו לב שאתם בודקים קריאה תקינה מהקובץ וערכי חזרה של הפונקציות שנקראות.
- עבור מספר ארגומנטים שגוי יש להדפיס מידע המסביר כיצד להריץ את הקוד (usage) ולצאת עם ערך חזרה שונה מ 0.
- מלבד ההנחות הרשומות אין להניח שהקלט תקין - עבור קלט שאינו תקין התוכנה לא אמורה לקרוס אלא להחזיר הודעת שגיאה.

2. <u>בדיקת התרגיל:</u>

- התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות, באמצעות קבצי קלט שונים (תרחישים שונים להרצת התכניות). הפלט של התוכנית שלכם יושווה (באמצעות השוואת

טקסט) לפלט של פתרון בית הספר. לכן עליכם להקפיד על פורמט הדפסה מדויק, כדי למנוע שגיאות מיותרות והורדת נקודות, ראו שימוש ב .diff

- אם ישנם מקרים שהוראות התרגיל לא מציינות בבירור כיצד התכנית צריכה להתנהג, הביטו בקבצי הקלט וקבצי הפלט לדוגמה שניתנים לכם ובדקו אם התשובה לשאלתכם נמצאת שם. כמו כן, היעזרו בפתרון בית הספר, הריצו עליו את הטסטים שלכם והשוו להתנהגות תוכניתכם. כמובן שניתן וכדאי להתייעץ בפורום לגבי מקרים שבהם התשובה עדיין אינה ברורה.

3. <u>חומר עזר:</u>

- אתם יכולים להיעזר באפליקציית java שגם מדפיסה את הטבלה. ניתן להוריד אותה מכאן: http://melolab.org/websoftware/web/?sid=3

ברצפים מסוג DNA אפשר להשתמש אך ורק באותיות ATCG
באופציית מטריצה לבחור ב identity. במקרה זה s ,1=m=0.
הערך של gap נקבע לפי פרמטר opening penalty.

- קבצי בדיקה לדוגמא ניתן למצוא ב:

```
~labc/www/ex2/files/
```

- מותר ואף רצוי להשתמש ב `diff` שבמחשבי האקוריום עבור השוואת פלטים (הסבר מפורט בסוף הקובץ).

4. <u>הגשה:</u>

- עליכם להגיש קובץ tar בשם `ex1.tar` המכיל רק את הקבצים הבאים:
  - `CompareSequences.c`

  ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

```
>tar -cvf ex2.tar CompareSequences.c
```

- <u>לפני ההגשה</u>, פתחו את הקובץ ex1.tar בתיקיה נפרדת וודאו שהקבצים מתקמפלים ללא שגיאות וללא אזהרות. וודאו שההגשה שלכם עוברת את ה-presubmission script ללא שגיאות או אזהרות.

```
~labc/www/ex2/presubmit_ex2
```

- אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:

```
~labc/www/codingStyleCheck <code file or directory>
```

  כאשר <directory or file> מוחלף בשם הקובץ אותו אתם רוצים לבדוק או תיקייה שיבדקו כל הקבצים הנמצאים בה (שימו לב שבדיקה אוטומטית זו הינה רק חלק מבדיקות ה codingStyle)

- <u>דאגו לבדוק לאחר ההגשה</u> את קובץ הפלט (submission.pdf) וודאו שההגשה שלכם עוברת את ה-presubmission script ללא שגיאות או אזהרות.

<u>שימוש בפקודת **diff:**</u>

- לרשותכם כמה קבצי קלט לדוגמה וקבצי הפלט המתאימים להם (אלו מהווים רק חלק קטן מקבצי הקלט-פלט שנשתמש בהם, כתבו לעצמכם בדיקות נוספות). עליכם לוודא שהתכנית שלכם נותנת את אותו הפלט בדיוק.

- על מנת לעשות זאת הריצו את תכניתכם עם הקלט לדוגמה על ידי ניתוב ה standard input להקרא מקובץ (באמצעות האופרטור "<" בשורת ההרצה ב terminal), ונתבו את הפלט של תכניתכם, שהוא ה standard output,לתוך קובץ (באמצעות האופרטור ">") באופן הבא:

```
prog_name < in_file > out_file
```

- השוו את קובץ הפלט שנוצר לכם עם קובץ הפלט המתאים של פתרון בית הספר, באמצעות הפקודה diff להשוואת טקסטים.

תיאור diff: בהינתן שני קבצי טקסט להשוואה (txt ,2.txt.1) הפקודה הבאה תדפיס את השורות אשר אינן זהות בשני הקבצים:

```
diff 1.txt 2.txt
```

במידה והקבצים זהים לחלוטין, לא יודפס דבר.

קראו על אפשרויות נוספות של diff בעזרת הפקודה man diff. לחלופין אתם יכולים גם להשתמש בתוכנה tkdiff אשר מראה גם את השינויים ויזואלית.

כמו כן, אתם יכולים גם להשוות ישירות באופן הבא:

```
prog_name < in_file | diff expected.out
```

בהצלחה!