234114-7 מבוא למדעי המחשב, סמסטר אביב 2020

תרגיל בית 4

מועד אחרון להגשה: **23:59 01.01.2020**

המתרגל האחראי על תרגיל זה: **קטרין חדאד**
E-mail: Catherine@cs.technion.ac.il
שעת קבלה: יום ב' 16:30-17:30

הנחיות:

- הגשה ב**בודדים**. עליכם לכתוב את הפתרונות לבד ולהגיש ביחידים.
- קראו את השאלות בעיון לפני שתתחילו בפתרונן.
- הקפידו לתעד את הקוד שלכם בהערות באנגלית.
- מלבד מילואים, לא יתקבלו תרגילים אחרי מועד הגשה. הגשה באיחור לאחר מועד הגשה נחשבת כאי-הגשה.
- כל יום מילואים = יום דחייה. על מנת לקבל את הדחייה, עליכם לשלוח באי-מייל למתרגל האחראי עותק של האישור המראה שהייתם במילואים (טופס 3010). אם האישור יגיע אליכם בתאריך מאוחר, יש להודיע על כך למתרגל האחראי על התרגיל.
- **לא ניתן לערער על תוצאות הבדיקה האוטומטית.**
- **שימו לב! הבדיקה הינה אוטומטית, ולכן הקפידו להדפיס בדיוק בפורמט שהתבקשתם ובידקו עם אתר הבדיקה ועם DiffMerge את הפלט שלכם מול הפלט של הדוגמאות שקיבלתם.**
  - השתמשו ב-redirection כדי להפנות את הפלט לקובץ טקסט.
  - וודאו את האותיות הגדולות והקטנות לפי הדוגמאות וההסברים בתרגיל.
  - אין להדפיס רווחים שלא התבקשתם להדפיס (בתחילת שורה או בסופה).
- בתרגיל זה מותר להשתמש בפונקציות מהספרייה stdio.h, stdlib.h, stdbool.h string.h למעט במקרים בהם נאמר אחרת. החומר הנדרש לתרגיל זה שייך לתרגולים 1-9. אין להשתמש בחומר שאינו מופיע במצגות אלה.
- ההגשה הינה אלקטרונית ו**בבודדים** דרך אתר הקורס. קובץ ההגשה יהיה מסוג **zip** (ולא אף פורמט אחר) ויכיל בתוכו את הקבצים הבאים בלבד, ללא כל תיקיות:
  - קובץ **students.txt** עם שמך **באנגלית**, מספר תעודת הזהות וכתובת האי-מייל שלך.
  - קובץ פתרון **hw4q1.c** עבור שאלה 1.
  - קובץ פתרון **hw4q2.c** עבור שאלה 2.
  - קובץ פתרון **hw4q3.c** עבור שאלה 3.
- **חובה לשמור את קוד אישור ההגשה שמקבלים מהמערכת לאחר שמגישים, עד לסיום הקורס.**
- יש להקפיד להגיש את כל הקבצים בדיוק עם השמות שמופיעים לעיל. הגשה שלא תעמוד בתנאי זה **לא תתקבל ע"י המערכת!** אם המערכת לא מקבלת את התרגיל שלכם, חפשו את הפתרון לבעיה באתר הקורס תחת הכפתור FAQ.

שאלה מספר 1

הקדמה:

תמונה במחשב מיוצגת כטבלה דו-מימדית (מטריצה) של ערכים. כל תא מכונה pixel (picture element). תמונה שמכילה בכל תא מספר שלם אי-שלילי מייצגת תמונת Grayscale ("שחור-לבן"), ואילו עבור ייצוג תמונת צבע נדרשת עבור כל פיקסל שלישיית ערכים, אשר לעיתים קרובות מייצגים את רכיבי האדום, הירוק והכחול – אולם קיימים גם ייצוגים אחרים.

בשאלה זו נממש שיטה שימושית לביצוע חישובים מהירים בתמונות: **טבלת שטח מסוכם**.

בהינתן תמונת grayscale בגודל n x m (n שורות על m עמודות), טבלת השטח המסוכם שלה מכילה בתא (i,j) את סכום כל התאים משמאלו ומעליו, וכן התא עצמו.

פורמלית, נסמן את התמונה המקורית כ-I ואת טבלת השטח המסוכם כ-I_Σ אז:

$$I_\Sigma[i][j] = \sum_{k=0}^{i}\sum_{l=0}^{j} I[k][l]$$

הערה: בתמונות מקובלת שהתא השמאלי העליון הינו I[0][0], האינדקס הראשון מציין מספר שורה, והאינדקס השני מציין מספר עמודה. למשל ערך התא I[2][0] בדוגמא הבאה הוא 4 – ערך התא הראשון בשורה השלישית.

למשל עבור התמונה הבאה:

I =

1	6	8	3
0	0	3	7
4	7	8	8
5	0	9	9

טבלת השטח המסוכם תהיה:

I_Σ =

1	7	15	18
1	7	18	28
5	18	37	55
10	23	51	78

לאחר שטבלת השטח המסוכם חושבה, נוכל להשתמש בה על מנת לקבל בחישוב פשוט סכום של ערכי הפיקסלים (בתמונה המקורית) בחלון מלבני מגודל כלשהו. עבור חלון הנתון ע"י האינדקסים:

$$Rect = (i_{top}, j_{left}, i_{bottom}, j_{right})$$

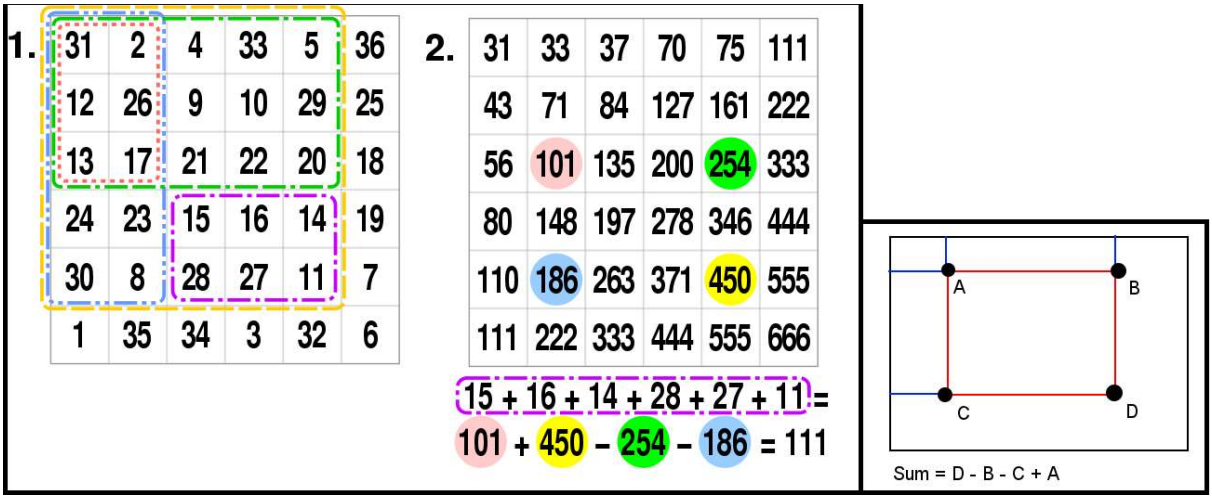
הנוסחא לסכום ערכי הפיקסלים היא:

$$\Sigma_{Rect} = I_\Sigma[i_{bottom}][j_{right}] - I_\Sigma[i_{top}-1][j_{right}] - I_\Sigma[i_{bottom}][j_{left}-1] + I_\Sigma[i_{top}-1][i_{left}-1]$$

למשל, הסכום של הריבוע המסומן בצהוב בדוגמא לעיל (Rect=(1,1,3,2)) מחושב ע"י:

$$\Sigma_{Rect} = I_\Sigma[3][2] - I_\Sigma[0][2] - I_\Sigma[3][0] + I_\Sigma[0][0] = 51 - 15 - 10 + 1 = 27$$

לדוגמאות נוספות מ-wikipedia על החישוב:



חלק א'

ממשו פונקציה עם החתימה:

```
void compute_integral_image(int image[][M], int n, int m, int integral_image[][M])
```

הפונקציה מקבלת תמונת grayscale בגודל n x m (כמו קודם, n - מספר השורות, m – מספר העמודות) ומחשבת את טבלת השטח המסוכם עבורה לתוך integral_image.

- הניחו שאורך השורה המירבי הינו 50 פיקסלים, והגדירו את הקבוע M בהתאם. באופן דומה, הניחו שאורך העמודה המירבי הינו 50 פיקסלים, והגדירו את הקבוע N בהתאם.
- הניחו שהקלט תקין, ז"א ש-image ו-integral_image מצביעים חוקיים לזיכרון מוקצה בגודל המתאים ו-n ו-m גדלים חוקיים שאינם חורגים מגודל הזיכרון המוקצה.
- **סיבוכיות**: על הפונקציה לעבוד בסיבוכיות זמן O(n*m) (לינארית בגודל התמונה) וסיבוכיות זיכרון נוסף O(1).
  - **רמז**: אם נוכל לחשב כל איבר במספר קבוע של פעולות (O(1)), נוכל לעמוד בדרישות הסיבוכיות (כי יש n*m איברים).לשם כך, חישבו מה הקשר בין האיבר ה-(i,j) ב-integral_image (אותו אנו רוצים לחשב בזמן קבוע) לבין האיברים ה-(i-1,j), (i,j-1) ו-(i-1,j-1) ב-integral_image.

חלק ב'

ממשו פונקציה עם החתימה:

```
#define RECT 4
int sum_rect(int integral_image[][M], int rect[RECT])
```

הפונקציה תחשב את Σ_{rect} לפי הנוסחא מעלה. גם כאן הניחו שהקלט תקין, בפרט rect מציין חלון מלבני בתמונה בדומה לדוגמא לעיל:

$$Rect = (i_{top}, j_{left}, i_{bottom}, j_{right})$$

- **סיבוכיות**: על הפונקציה לעבוד בסיבוכיות זמן O(1) וסיבוכיות זיכרון נוסף O(1).

חלק ג'

ממשו פונקציה עם החתימה:

```
void sliding_average(
   int integral_image[][M], int n, int m, int h, int w, int average[][M]
)
```

הפונקציה מקבלת את טבלת השטח המסוכם - integral_image, בגודל n x m, ומחשבת לתוך average ממוצע על פני חלון נע בגודל h x w של הערכים בתמונה המקורית, כלומר הערך באיבר ה-(i,j) ב-average (כלומר average[i][j]) יהיה הערך הממוצע במלבן בתמונה המקורית שמרכזו בפיקסל (i,j) וגודלו h x w.

- הגודל של המטריצה average כגודל תמונת הקלט.
- כמו כן, הניחו שמידות החלון אי-זוגיות (גם h וגם w), וגם שהינן קטנות מגודל התמונה.
- עבור הערכים בקצוות (כאשר המלבן "לא נכנס" כולו בתמונה. כלומר "גולש" מחוץ לתמונה), הניחו שמחוץ לתמונה הערכים הם 0 (מה זה אומר על הערכים בטבלת השטח המסוכם שמחוץ לתחום?). למשל עבור מלבן 3 x 3 והדוגמא לעיל, מתקיים:

average[0][1] = (0 + 0 + 0 + 1 + 6 + 8 + 0 + 0 + 3) / 9 = 2

הערה: כידוע, בפועל בשפת סי הערכים מחוץ למטריצה שיצרנו אינם בהכרח 0, עליכם לדאוג למקרים אלה במיוחד.

טבלת הערכים המלאה:

1	2	3	2
2	4	6	4
2	4	6	5
2	4	5	4

average =

פורמלית, המטריצה average מחושבת לפי (כאשר image היא התמונה המקורית):

$$average[i][j] = \frac{1}{h \cdot w} \sum_{k=-\lfloor \frac{h}{2} \rfloor}^{\lfloor \frac{h}{2} \rfloor} \sum_{l=-\lfloor \frac{w}{2} \rfloor}^{\lfloor \frac{w}{2} \rfloor} image[i+k][j+l]$$

(כאשר ההנחה שהערכים מחוץ ל-image הם 0).

שימו לב, מכיוון ש-average מכילה מספרים שלמים, את הממוצע יש לחשב בעיגול למספר השלם הקרוב ביותר (לשם כך יש להשתמש בפונקציה round בספריית math.h). שימו לב שכאשר אתם מחשבים את הממוצע אתם לא מבצעים חילוק שלמים בטעות.

- **סיבוכיות**: על הפונקציה לעבוד בסיבוכיות זמן O(n*m) וסיבוכיות זיכרון נוסף O(1). כאשר n ו-m הם גודל התמונה שקיבלנו. שימו לב שלא צריכה להיות תלות בגודל החלון הנע! כדי שנוכל לממש פונקציה זו בסיבוכיות הנדרשת, עלינו להיות מסוגלים לחשב כל איבר בזמן קבוע O(1). לשם כך, עשו שימוש בפונקציה sum_rect, ב-integral_image ובנוסחא מההקדמה. שימו לב שיש לטפל במקרים של חריגה מגבולות התמונה.
- **רמז**: אנו רוצים לעשות שימוש ב-integral_image ובנוסחא מההקדמה. חישבו מה קורה כאשר גולשים\חורגים מגבולות התמונה לשמאל ו\או למעלה. מה קורה כאשר גולשים לימין ו\או למטה. שני המקרים שונים.

הנחיות לפתרון התרגיל:

- באתר הקורס מסופק לכם קובץ המכיל את פונקציית ה- main ופונקצית ההדפסה. יש להוריד את הקובץ ולממש את כל הפונקציות לעיל בקובץ זה.
- לאורך התרגיל הניחו שהקלט מהמשתמש חוקי, ז״א כאשר נדרש – המשתמש מזין מימדי תמונה תקינים (n,m >=1), תמונה בגודל המתאים עם ערכים תקינים עבור הפיקסלים וכן מידות תקינות לחלון נע – ז״א מצבים לא תקינים לא יינתנו כמקרי קלט, ולכן הפלט המדויק במקרים אלו אינו משנה. יחד עם זאת, עליכם להקפיד על כללי תכנות נכון כפי שנלמדו בכיתה, להקפיד על חלוקה הגיונית לפונקציות, להקפיד על קריאות הקוד, שמות משמעותיים למשתנים ופונקציות, הגדרת קבועים וכו׳.
- מומלץ לפתור באמצעות עקרונות top-down design. חלקו את העבודה לתתי-משימות, ובכל פעם ממשו תת-משימה אחרת, בדיוק כמו שלמדנו.

דוגמת הרצה:

```
Enter image dimensions:
4 5
Enter image:
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
Enter sliding window dimensions:
3 3
Integral image is:
1 3 6 10 15
7 16 27 40 55
18 39 63 90 120
34 72 114 160 210
Smoothed image is:
2 3 4 4 3
4 7 8 9 6
8 12 13 14 10
6 10 10 11 8
```

דוגמאות נוספות בקבצי הקלט\פלט המסופקים לכם.

שאלה מספר 2

שאלה זו עוסקת במחרוזות ובהקצאת זיכרון דינאמית

בשאלה אתם מתבקשים לממש צופן הודעות בשפה האנגלית.

אופן פעולת הצופן:

- **הודעה**: היא רצף של מילים בשפה האנגלית המופרדים ע"י רווח. כל המילים בהודעה מורכבים מאותיות אנגליות גדולות או קטנות אך אינם מכילים מספרים או תווים מיוחדים. בכל הודעה ישנם **עד** 100 מילים.
- **מפתח**: מפתח ההצפנה הוא התאמה בין אות בהודעה המקורית לתו אחר בהודעה המוצפנת. בתרגיל זה אנו נתאר מפתח ע"י מערך של תווים בן 52 תאים, שבהם רשומים כל האותיות האנגליות הגדולות והקטנות. כל אות מופיעה בדיוק פעם אחת. ב- 26 התאים הראשונים במערך מופיעה ההצפנה של האותיות A-Z לפי הסדר (כלומר בתא ה-0 רשומה ההצפנה של האות A ..), וב- 26 התאים הבאים אחריהם מופיעה ההצפנה של האותיות a-z.
- בקבלת הודעה כלשהי, עליכם לעבור מילה מילה, ולכל מילה להפוך את האותיות שלה לפי מפתח ההצפנה.
- באתר מסופק לכם קובץ hw4q2main.c המכיל את הפונקציה main ופונקצית ההדפסה. יש להוריד את הקובץ ולרשום לתוכו את המימוש שלכם. אין לשנות את המימוש של פונקצית ה-main או את המימוש של פונקצית ההדפסה. ניתן להניח שאורך כל אחת מהמילים בקלט הוא עד 50 תווים (לא כולל התו '\0').
- יש להשתמש במפתח המוגדר לכם בפונקצית ה- main.

חלק א'

ממשו את הפונקציה הבאה:

```
#define KEY_SIZE 52
char encrypt(unsigned char key[KEY_SIZE], char ch);
```

פעולת הפונקציה: הפונקציה מקבלת את המפתח, ותו להצפנה ch. הפונקציה צריכה להחזיר את התו שנמצא באינדקס המתאים במערך key.

חלק ב'

ממשו את הפונקציה הבאה:

לצורך סעיף זה, מילה היא אוסף של תווים שלא כולל את התו רווח.

```
int read_words(char* words[], int size, int max_str_len);
```

פרמטרים:

- words – מערך של מצביעים (המערך יעודכן ע"י הפונקציה, כלומר הפונקציה תכתוב למערך ערכים של מצביעים חדשים, כאשר כל מצביע יצביע למחרוזת עם מילה).
- size – אורך המערך words, במספר מצביעים.
- max_str_len – אורך מקסימלי של מילה בקלט.

פעולת הפונקציה: הפונקציה צריכה לקלוט מהקלט הסטנדרטי עד size מילים, להקצות באופן דינמי עבור כל מילה בלוק זיכרון שיוכל להכיל אותה, להעתיק אותה אליו ולשמור את כתובת המחרוזת במערך המצביעים words. (יתכן שהפונקציה תקלוט פחות מ size מילים, במידה וסוף הקלט כולל פחות מ- size מילים).

על מנת לחסוך בזיכרון, הקצאת הבלוקים של הזיכרון תיעשה בגודל המינימלי הדרוש, כלומר בהתאם לאורך המילים שנקלטו בפועל (יש לשים לב להקצות מקום גם לתו ה- \0).

ערך חוזר:

- במקרה ההצלחה: הפונקציה תחזיר את מספר המילים שנקלטו.
- במידה והקצאת זיכרון נכשלת, הפונקציה צריכה להחזיר 1-. כמו כן, הפונקציה צריכה לשחרר את כל הזיכרון שהיא הקצתה (במידה והיא הצליחה לבצע הקצאות זיכרון חלקיות).

דרישות סיבוכיות זמן: O(size*max_str_len),

סיבוכיות מקום: O(n+ max_str_len) כאשר n מסמן את מספר התווים שנקלטו (שיכול להיות קטן משמעותית מ size*max_str_len).

חלק ג'

ממשו את הפונקציה הבאה:

```
void encrypt_words(char* words[], int num_words,
                   unsigned char key[KEY_SIZE]);
```

הפונקציה צריכה להצפין את כל המחרוזות במערך words ע"י שימוש בפונקציית ההצפנה מסעיף א'. הפונקציה אמורה לשנות את המילים במערך המילים עצמו.

דרישות סיבוכיות זמן: O(n) כאשר n הוא סך אורכי המחרוזות במערך words.

דרישות סיבוכיות מקום: O(1).

חלק ד'

```
void release_memory(char* words, int size);
```

הפונקציה משחררת את כל הזיכרון שהוקצה במערך המילים.

דוגמאות הרצה:

input	**output**
if you want to shine like a sun first you need to burn like it	rc hod smtg go jqrtz eriz m jdt cr$jg hod tzzv go bd$t eriz rg
If you CANNOT STAND THE heat GET out of the KItChen	Oc hod E>FFGZ LZ>FR ZIT qzmg UTZ odg oc gqz ~OgEqzt

שאלה מספר 3

שאלה זו עוסקת בניתוח סיבוכיות

בשאלות הבאות בחרו בבקשה את האות המייצגת את הסיבוכיות המתאימה מתוך הרשימה הבאה:

a. $\theta(1)$	*n.* $\theta(n^2\sqrt{n})$
b. $\theta(logn)$	*o.* $\theta(\sqrt{3^n})$
c. $\theta(log^2 n)$	*p.* $\theta(nlog(logn))$
d. $\theta(\sqrt{n})$	*q.* $\theta(n^3 log^2 n)$
e. $\theta(\sqrt{n}logn)$	*r.* $\theta(2^n)$
f. $\theta(\sqrt{n}\, log^2 n)$	*s.* $\theta(n * 2^n)$
g. $\theta(n)$	*t.* $\theta(n^2 2^n)$
h. $\theta(nlogn)$	*u.* $\theta(n^3 2^n)$
i. $\theta(nlog^2 n)$	*v.* $\theta(4^n log\, n)$
j. $\theta(n\sqrt{n})$	*w.* $\theta(4^n log^2 n)$
k. $\theta(n^2)$	*x.* $\theta(n4^n)$
l. $\theta(n^2 log\, n)$	*y.* $\theta(n^2 4^n)$
m. $\theta(n^2 log^2 n)$	*z.* $the\ correct\ answer\ doesn't\ appear$

אופן הגשת השאלה: את שאלה זו יש להגיש באמצעות הקובץ hw4q3.c שנמצא באתר הקורס, עליכם לערוך את הקובץ בהתאם לתשובה שסימנתם עבור שאלות 1, 2, 3, 4, 5, 6.

קטע קוד מספר 1:

חשבו את סיבוכיות הזמן והמקום של הפונקציה f1:

```
void f1(int n)
{
    int b = 2;
    while (n >= 1)
    {
        b *= b;
        n /= 2;
    }
    free(malloc(b));
}
```

1. סיבוכיות זמן: _____

2. סיבוכיות מקום: _____ _____

קטע קוד מספר 2:

חשבו את סיבוכיות הזמן והמקום של הפונקציה f2.

```
void g2(int n) {
    while (n>0) {
        printf("*");
        n--;
    }
}

void f2(int n){
    int k = n/2;
    for (int i=2; i<n; i*=i) {
        g2(k);
    }
}
```

3. סיבוכיות זמן: _____

4. סיבוכיות מקום: _____

קטע קוד מספר 3:

חשבו את סיבוכיות הזמן והמקום של הפונקציה f2:

```
int f1(int n)
{
    int s=0;
    for(int i=0; i<n; i++)
        s+=i;
    return s;
}

void f2(int n)
{
    for(int i=n; i>0; i/=2)
    {
        int* p = (int*)malloc(n*sizeof(int));
        int k=f1(i);
        for(int j=0; j<k; j++)
            printf("Good Luck!");
        free(p);
    }
}
```

5. סיבוכיות זמן: _____

6. סיבוכיות מקום: _____

בהצלחה!