מבוא למדעי המחשב, סמסטר קיץ 2018

תרגיל בית 4

מועד אחרון להגשה: **יום ב' – 10.9.2017 23:59**

המתרגל האחראי על תרגיל זה: **עידו רפאל**

E-mail: ungulient@gmail.com (נא להוסיף את מספר הקורס –"234114" לכותרת המייל, כדי שהמייל יגיע לתיקייה הנכונה ולא אפספס אתכם)
שעת קבלה רגילה: אחרי התרגולים בימים ב' ו-ד' (התרגול נגמר ב-15:20. נא לא לאחר, כדי שלא אפספס אתכם).

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 על תרגיל זה 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 למעט במקרים בהם נאמר אחרת. החומר הנדרש לתרגיל זה שייך לתרגולים 0-9 (כלומר עד התרגול של Complexity, כולל). 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.

בדיקה ידנית:

בנוסף לבדיקה האוטומטית, התרגיל ייבדק גם בבדיקה ידנית. הבדיקה תתמקד בנושאים הבאים:

- אורך **מימוש** (לא כולל ההכרזה, או שם הפונקציה) כל פונקציה לא יעלה על 16 שורות קוד.

  הגבלה זו תקפה לכל הפונקציות, כולל main.

  רק שורות ריקות, שורות עם סוגר מסולסל **בלבד**, ושורות עם הערות **בלבד לא** נספרות.

  רווחים לבנים לא נספרים.

  אסור לכתוב כמה פקודות שונות משמעותית באותה השורה, למשל כותרת תנאי\לולאה

  צריכה להיות בשורה נפרדת מגוף התנאי\לולאה.

- רוחב שורה - רוחב כל שורה (**כולל הערות והזחות**) לא יעלה על **80** תווים.

ניתן לסמן אורך של שורה בקודבלוקס:

Code::Blocks -> Settings -> Editor -> Margins and caret ->

Right margin hint: Visible line, **Hint column**: 80

- קבועים בקוד:
  - יש להגדיר בעזרת #define כל קבוע משמעותי. שמות קבועים צריכים להיות באותיות גדולות בלבד, אם השם מכיל יותר ממילה אחת מילים יופרדו בעזרת מקף תחתון (למשל STUDENTS_NUM).
  - אין להשתמש בערכי ASCII ישירות. יש להשתמש בייצוג של התווים (למשל 'a').
- הזחות:
  - שיטת הזחה מקובלת – הזחת קוד בכל בלוק, למשל:

```
int main()
{
	// your code here
	while (…)
	{
		// your code here
	}
}
```

- אין להשתמש במשתנים גלובאליים או סטטיים.
- שמות משתנים/קבועים/פונקציות צריכים להיות אינפורמטיביים, להעיד על מטרתם.
- בהירות הקוד ותיעוד:
  - יש לתעד את הקוד באמצעות הערות באנגלית בלבד. במידה ויש כמה שורות קוד שניתן להסביר בקצרה מה המטרה שלהן, יש לשים הערה בהתחלה ואין צורך לתעד כל שורה.
  - יש לתעד פונקציות – לפני הפונקציה להוסיף הערה שמסבירה בקצרה מה הפונקציה עושה ומה המשמעות של הפרמטרים שלה.
  - התיעוד צריך להיות אינפורמטיבי, כלומר יש להסביר מה המטרה של שורות הקוד ולא לכתוב את הקוד במילים.
- שכפול קוד שלא לצורך (למשל ריבוי קוד זהה במספר מקרי if-else שונים).
- אי-עמידה באחת מדרישות התרגיל (שימוש בחומר שהיה אסור בתרגיל וכו').

באופן כללי – הקפידו על כתיבת קוד מסודר ומובן ככל שניתן תוך יישום העקרונות שנלמדו בכיתה.
מותר לכם לממש פונקציות עזר משלכם ולהשתמש בהן.

שאלה 1: עיבוד תמונה

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:

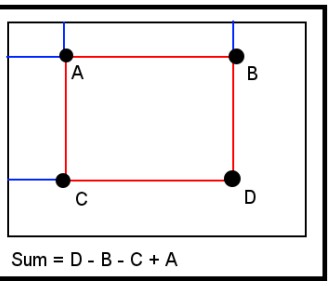
1.

31	2	4	33	5	36
12	26	9	10	29	25
13	17	21	22	20	18
24	23	15	16	14	19
30	8	28	27	11	7
1	35	34	3	32	6

2.

31	33	37	70	75	111
43	71	84	127	161	222
56	101	135	200	254	333
80	148	197	278	346	444
110	186	263	371	450	555
111	222	333	444	555	666

15 + 16 + 14 + 28 + 27 + 11 =
101 + 450 – 254 – 186 = 111



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.

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.
כמו כן, שימו לב מה קורה כאשר חורגים מגבולות התמונה, ואז תשכחו לטפל במקרה זה.

חלק ב'

ממשו פונקציה עם החתימה:

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

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 ב**תמונה המקורית**, כלומר הערך באיבר ה-(i,j) ב-average (כלומר average[i][j]) יהיה הערך הממוצע במלבן ב**תמונה 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:

average =

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

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

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

(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.

<u>חלק ד'</u>

כתבו תכנית שקולטת מהמשתמש תמונה ומידות חלון נע, ומשתמשת בפונקציות מהסיעיפים הקודמים על מנת לפלוט את טבלת השטח המסוכם וממוצע על פני חלון נע (תמונת ממוצעים). הניחו כי גודל התמונה לא יעלה על 50 x 50 פיקסלים.

כדי להבין מה בדיוק פורמט הפלט המצופה מכם (למשל מתי לשים רווח ומתי לרדת שורה), בדקו את קבצי הקלט\פלט המסופקים לכם.

דוגמא להרצת התכנית:

```
Enter image dimensions:
4 4
Enter image:
1 6 8 3
0 0 3 7
4 7 8 8
5 0 9 9
Enter sliding window dimensions:
3 3
Integral image is:
1 7 15 18
1 7 18 28
5 18 37 55
10 23 51 78
Smoothed image is:
1 2 3 2
2 4 6 4
2 4 6 5
2 4 5 4
```

דוגמאות נוספות בקבצי הקלט\פלט המסופקים לכם.

הנחיות לפתרון התרגיל:

- 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, לוודא תקינות קלט (למרות שלא ייבדק בבדיקה האוטומטית), להקפיד על חלוקה הגיונית לפונקציות, להקפיד על קריאות הקוד, שמות משמעותיים למשתנים ופונקציות, הגדרת קבועים וכו'.
- מומלץ לפתור באמצעות עקרונות top-down design. התחילו את הפתרון מסעיף ד'. הניחו שגודל התמונה לא יעלה על 50 x 50 פיקסלים, והקצו את התמונה וטבלת השטח המסוכם כמטריצות דו-מימדיות בגודל המתאים. הגדירו פונקציות ריקות (או עם מימוש נאיבי) עם החתימות הדרושות, וכן פונקציות נוספות לפי הצורך. חלקו את העבודה לתתי-משימות, ובכל פעם ממשו תת-משימה אחרת, בדיוק כמו שלמדנו.

שאלה 2: מחרוזות

בשאלה זו נרצה לממש מנוע חיפוש מבוסס ASCII – המשתמש יקליד את מספר המחרוזות אותן הוא מעוניין לחפש, לאחר מכן יקליד את המחרוזות והתכנית תפלוט מחרוזות הנמצאות במילון, השקולות בערכן ה-ASCII למחרוזת שהוכנסה.

אנחנו נתייחס לערך ה-ASCII הפשוט של מחרוזת. כלומר ערך ה-ASCII של מחרוזת שווה לסכום ערכי ה-ASCII של התווים המרכיבים אותה.

לדוגמא, ערך ה-ASCII של המחרוזת "BRAZIL" הינו:
ASCII_Value("BRAZIL") = 'B' + 'R' + 'A' + 'Z' + 'I' + 'L' = 66 + 82 + 65 +90 +73 +76 = 452

נממש את מנוע החיפוש:

1. יצירת מילון מחרוזות המכיל את המחרוזות הבאות:

```
"AVOCADO", "BANNANA", "WATER", "CAT", "DOG", "HELLO",
"ISRAEL", "ITALY", "GARMISH", "USA", "BRAZIL", "KING", "ABCDy"
"INT", "CHAR", "DOUBLE", "hardcheese", "realize", "farther"
```

2. נבקש מהמשתמש להקיש מספר שלם ואי שלילי, אשר יסמן כמה מחרוזות הוא רוצה לחפש, ע"י הדפסת ההודעה:

```
"Please enter a non-negative number:\n"
```

- ניתן להניח כי המספר שלם.
- יש לבדוק כי המספר אי שלילי. כל עוד אינו אי שלילי, יש לבקש מהמשתמש להכניס מספר חדש ע"י הדפסת ההודעה:

```
"Please enter a NON-NEGATIVE number:\n"
```

- חישבו על מימוש פשוט ע"י לולאת do-while, והדפסה בעזרת printf והדגל %s.

3. נבקש מהמשתמש להכניס את מספר המחרוזות אותו הוא ביקש לחפש ע"י הדפסת ההודעה:

```
"Please enter %d strings:\n"
```

- **אין** לטפל במיוחד במקרה של מחרוזת אחת (כלומר עדיין יש להדפיס "1 strings").
- **יש** לטפל במיוחד במקרה של 0 מחרוזות, ולהדפיס את ההודעה:

```
"no strings\n"
```

4. המשתמש יכניס את המחרוזות (ניתן להניח שמופרדות בירידת שורה וכן כי לא מכילות רווחים\whitespaces. כלומר ניתן לקלוט ע"י scanf ו-%s ולא לדאוג שתתעלם מרווחים).
לכל מחרוזת (ניתן להניח שבגודל 10 לכל היותר):
   - במידה ונמצאו התאמות (כל מחרוזת עם ערך ASCII שווה לערך ה-ASCII של המחרוזת שלנו):
   נדפיס:

```
"the string %s has %d matches in the dictionary:\n"
```

   כאשר %s היא המחרוזת, ו-%d הוא מספר ההתאמות שמצאנו במילון.
   ומתחת להודעה זו, נדפיס את כל המחרוזות מהמילון שערך ה-ASCII שלהן שווה לערך ה-ASCII של המחרוזת שלנו. אם יש יותר מאחת, סדר ההדפסה הוא לפי סדר הופעת המחרוזות במילון.

כמו כן, יש להפריד בין המחרוזות שנמצאו מתאימות ע"י ירידת שורה אחת בין מחרוזת למחרוזת (וכמובן גם ירידת שורה אחרי המחרוזת האחרונה, ולא רק ביניהן).

- במידה ולא נמצאו התאמות:
נדפיס:

```
"the string %s has no matches in the dictionary\n"
```

כאשר %s היא המחרוזת שלנו.

- לאחר **כל איטרציה** של הלולאה הראשית שמוגדרת בסעיף 4, יש להדפיס בנוסף **2** ירידות שורה.

דוגמאות הרצה:

```
Please enter a non-negative number:
1
Please enter 1 strings:
example
the string example has 2 matches in the dictionary:
realize
farther
```

```
Please enter a non-negative number:
0
no strings
```

```
Please enter a non-negative number:
-6
Please enter a NON-NEGATIVE number:
3
Please enter 3 strings:
WATER
the string WATER has 3 matches in the dictionary:
WATER
ITALY
ABCDy


ITALY
the string ITALY has 3 matches in the dictionary:
WATER
ITALY
ABCDy


hsercdhee
the string hsercdhee has no matches in the dictionary
```

לדוגמאות נוספות, בדקו את קבצי הקלט\פלט המסופקים לכם.

שאלה 3: ניתוח סיבוכיות

כל הסעיפים דורשים סיבוכיות זמן ב**מקרה הגרוע**.

בשאלות 0-5 (**בעמוד הבא**) בחרו את האות המייצגת את הסיבוכיות המתאימה מתוך טבלת הפונקציות להלן:

a. $\Theta(1)$

b. $\Theta(\log n)$

c. $\Theta(\log^2 n)$

d. $\Theta(\sqrt{n})$

e. $\Theta(\sqrt{n}\log n)$

f. $\Theta(\sqrt{n}\log^2 n)$

g. $\Theta(n)$

h. $\Theta(n\log n)$

i. $\Theta(n\log^2 n)$

j. $\Theta(n\cdot\sqrt{n})$

k. $\Theta(n^2)$

l. $\Theta(n^2\log n)$

m. $\Theta(n^2\log^2 n)$

n. $\Theta(n^2\cdot\sqrt{n})$

o. $\Theta(n^3)$

p. $\Theta(n^3\log n)$

q. $\Theta(n^3\log^2 n)$

r. $\Theta(2^n)$

s. $\Theta(n\cdot 2^n)$

t. $\Theta(n^2\cdot 2^n)$

u. $\Theta(n^3\cdot 2^n)$

v. $\Theta(3^n)$

w. $\Theta(n\cdot 3^n)$

x. $\Theta(n^2\cdot 3^n)$

y. $\Theta(n^3\cdot 3^n)$

z. *the correct answer doesn't appear*

השאלות:

```
void f1(int n) {
    int s = 1;
    for (int i = 0; i < n; i++) {
        s *= 2;
        for (int j = 0; j < s * n; j++) {
            printf("hi!\n");
        }
    }
}
```

0. מה היא סיבוכיות הזמן של הפונקציה **f1 כתלות ב-n**?
1. מה היא סיבוכיות המקום של הפונקציה **f1 כתלות ב-n**?

```
#define K 5

int f2(int n) {
    int s = 0;
    int d, m;

    while (n > 1) {
        scanf("%d", &m);
        d = (m > 2 * K) ? 2 * K : (m < K ? K : m);
        n = n / d;
        s += n;
    }
    return s;
}
```

2. מה היא סיבוכיות הזמן של הפונקציה **f2 כתלות ב-n**?
3. מה היא סיבוכיות המקום של הפונקציה **f2 כתלות ב-n**?

```
int f3(int k) {
    int s = k;
    for (int m = 2; k > 0; k /= m) {
        s = s + k;
    }
    return s + 1;
}

void f4(int n) {
    for (int j = 0; j < n * f3(n); ++j){
        for (int i = 1; i * i < n; i *= 2){
            printf("Hello j!\n");
        }
    }
}
```

4. מה היא סיבוכיות הזמן של הפונקציה **f4 כתלות ב-n**?
5. מה היא סיבוכיות המקום של הפונקציה **f4 כתלות ב-n**?

אופן הגשת התשובות בשאלה 3:

מסופק לכם קובץ שלד **hw4q3.c** אשר נראה כך:

```
int main()
{
    int question;
    char answer[] = {
        'a', 'b', 'c', 'd', 'e', 'f'
    };

    scanf("%d",&question);

    printf(
       "The answer to question %d is %c\n",
       question, answer[question]
    );

    return 0;
}
```

עליכם לערוך את התשובות במערך answer בהתאם.

למשל, בקובץ השלד, התשובות לשאלות 0-5 הן a-f בהתאמה, כלומר לשאלה 0 התשובה היא a, לשאלה 1 התשובה היא b, לשאלה 2 התשובה היא c וכו'. עליכם לשנות את התשובות במערך לתשובות ה**נכונות**.

קלט\פלט צפויים:

כפי שניתן לראות מקובץ השלד, הקלט לתרגיל זה יהיה מספר השאלה (0 עד 5).

סיפקנו לכם קבצי **קלט** של שאלות 0-5 (שבסה"כ מכילים את מספר השאלה וירידת שורה), אך כמובן שלא סיפקנו קבצי **פלט** לדוגמא (ולכן גם לא באתר), מכיוון שאתם בסה"כ מתבקשים להחליף את הערכים שיש במערך, ואחרת הייתם יכולים לדעת את התשובות בלי לפתור את השאלות 😊.