מבוא לתכנות מערכות 10010 סמסטר ב' – תש"פ

תרגיל בית מס' 1

נושא התרגיל: מטריצות מצביעים

יש להגיש אך ורק דרך תפריט המטלות שבאתר הקורס, כפי שהוסבר בתרגול . **ניתן לעבוד בזוגות**

הנחיות הגשה כלליות:

- התרגיל ייבדק בסביבת.Ubuntu
- הקוד חייב לעבור קומפילציה, קוד שאינו מתקמפל לא ייבדק.
- **יש להגיש קוד ללא הערות קומפילציה, warnings, קוד בו יהיו הערות יגרור הורדה של 10 נקודות.**
- יש להגיש את תיקיית הקוד, מקובצת לקובץ אחד ששמו כשם הסטודנט. שם פרטי ומשפחה
- במידה ומגישים בזוג יש לכתוב את שמות 2 המגישים ושני המגישים צריכים לעלות את העבודה למודל.

הוראות כלליות:

1. יש להקפיד על כללי הנדסת התוכנה:
   1. מבנה התכנית (הזחות) ותיעוד במידת הצורך .
   2. חובה להשתמש בקבועים במקומות המתאימים .
   3. יש להשתמש בפונקציות קצרות, כלליות, קריאות ושימושיות.
   4. יש להקפיד על בדיקת תקינות קלט, אפשר להניח שאם ביקשו מהמשתמש מספר הוא הכניס מספר אך יתכן ולא בטווח הנכון.
   5. הפלט צריך להיות כפי שניתן בתרגיל.
   6. קוד קצר, לא מסורבל ויעיל הן מבחינת כתיבתו והן מבחינת ריצת התוכנית.

1. יש לחלק את הפרויקט לקבצים. כל משימה תהיה בזוג קבצים *.c ו *.h בנוסף יהיו עוד זוג קבצים general.c ו general.h שבהם יהיו פונקציות כלליות או שנדרשות בכמה מהמשימות
2. קובץ main.c ניתן לכם.
3. יש ליצור ולהגיש **Makefile** מתאים לפרויקט כפי שנלמד בכיתה.
4. יש לקרוא את ההוראות במדויק, יש אילוצים מסוימים בכל משימה.
5. **שים לב שיש לחלק לפונקציות ורק הפונקציה הראשית בכל משימה תדפיס תוצאות.**

פירוט תרגיל בית 1

בתרגיל זה 3 משימות שונות. התוכנית הראשית תציג תפריט התחלתי המאפשר למשתמש לבחור משימה או לצאת ואז נברך אותו לשלום.

התפריט הראשי

```
Enter your choice of exe
1 – Biggest Matrix Sum
2 – Squares
3 – Lines
-1 -  Exit
```

נתון ב main עם switch case ראשי עם קריאה לפונקציה ראשית של כל תרגיל.

```
#define BIG_SUM 1
#define SQUARE  2
#define LINES   3
#define EXIT  -1

int menu();

int main()
{
	int option;
	int stop = 0;
	srand((unsigned)time(NULL));
	do
	{
		option = menu();
		switch (option)
		{
		case BIG_SUM:
			findMaxSubMatrix();
			break;

		case SQUARE:
			pictureSquares();
			break;

		case LINES:
			pictureLines();
			break;

		case EXIT:
			printf("Bye bye\n");
			stop = 1;
			break;

		default:
			printf("Wrong option\n");
			break;
		}
	} while (!stop);
	return 1;
}
int menu()
{
	int option;

	printf("\n\n");
	printf("Please choose one of the following options\n");
	printf("%d - Biggest Matrix Sum\n", BIG_SUM);
	printf("%d - Squares\n",SQUARE);
	printf("%d - Lines\n",LINES);
	printf("%d - Quit\n",EXIT);

	scanf("%d", &option);

	return option;
}
```

משימה 1 - Biggest Matrix Sum:

במשימה זו נחפש תת מטריצה בתוך מטריצה גדולה שסכום אבריה המקסימאלי.

- דוגמא: עבור מטריצה 20X20 אשר מחפשים בה תת מטריצה בגודל 2X2 נמצאה המטריצה המודגשת בצהוב. **שים לב: הדוגמא אומנם מתייחסת למטריצות מרובעות אך הדבר לא מחויב.**

50	50
50	49

23	35	40	-2	1	3	-28	-34	-45	46	-16	35	-21	29	-22	-24	17	-7	-4	13
-13	-20	11	21	-31	43	8	17	-40	0	47	-15	9	-45	47	13	11	22	-47	-36
34	4	-45	-12	-28	-10	36	32	-24	27	-49	50	50	-18	21	27	4	-15	-22	40
-11	-20	43	-39	13	-4	33	25	7	44	24	-24	-4	-13	-4	8	26	2	19	-16
18	-24	42	-30	15	-4	-23	19	-42	-10	-47	-44	-19	47	-45	-19	-32	15	8	20
-30	13	42	21	-35	-25	-21	13	-47	36	-26	6	16	48	24	43	40	-5	36	23
-48	48	-25	3	32	-36	3	-46	50	50	32	-25	33	-47	48	42	21	-37	-42	-46
-4	46	36	20	-36	48	50	4	50	49	14	19	-18	-21	10	-22	-2	-45	-6	15
-18	17	3	42	-24	-37	7	4	-46	-6	12	41	11	-43	17	-2	-26	17	38	38
50	7	-22	40	-20	-21	21	-19	3	46	35	1	-26	-20	23	19	20	28	-25	-3
-17	21	-9	20	28	21	-12	5	38	43	9	-27	25	-49	37	17	22	-21	-46	19
10	-19	-6	16	34	-32	-15	-46	24	-12	5	41	-10	35	21	-44	-13	18	16	-42
-37	14	12	-36	8	23	9	26	-15	0	-6	22	-32	-4	13	-43	21	9	-35	27
-24	-41	42	1	37	-1	34	48	30	0	9	-6	-44	31	10	-37	28	16	46	-9
20	16	-23	8	33	35	-41	-8	-42	11	19	18	45	36	32	-24	37	24	21	36
-46	38	-9	28	40	11	19	50	-20	50	-19	24	3	20	47	9	21	7	-17	-24
49	-17	2	40	2	-29	3	-22	16	40	3	41	43	-46	1	-17	4	6	44	-22
-25	20	5	50	1	-2	-26	40	10	4	-50	-29	18	49	-30	6	-43	-48	-18	-28
-25	15	-27	-22	46	19	17	45	49	43	15	-31	-31	-23	-24	-27	-3	-29	-26	19
-40	17	-20	-13	-29	-49	25	24	-3	-5	48	13	39	14	30	47	44	-45	30	-7

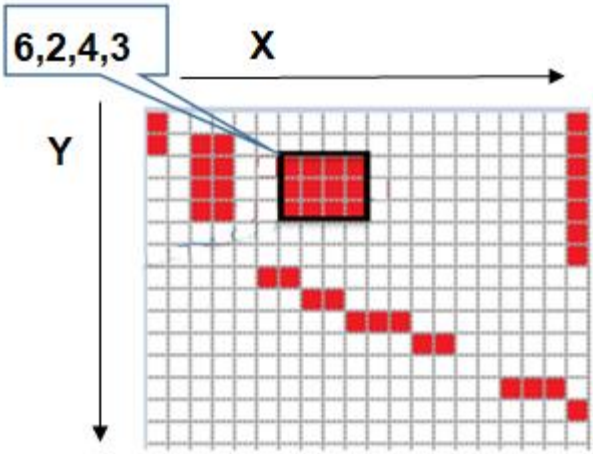
המשימה מבצעת את הפעולות הבאות: שים לב כל שלב הוא לפחות פונקציה אחת נפרדת

1. הגדר מטריצה בגודל ROWS שורות ו COLS עמודות בחר ערכים כרצונך. *הדפס ערכים אילו למסך.*
2. קלוט מהמשתמש שני מספרים אשר יגדירו את גודל המטריצה האפקטיבי כך שמספר השורות יהיה בין 1 ל ROWS כולל ומספר העמודות יהיה בין 1 ל COLS כולל. *הדפס ערכים אילו למסך.*
3. צור את המטריצה

משלב זה אין להשתמש ב סוגריים [ ] ואין להגדיר עוד מטריצה מלבד המטריצה הגדולה.

4. אתחל את המטריצה במספרים אקראיים, בחר טווח כרצונך, הגדר כמובן בקבועים. *הדפס ערכים אילו למסך.*
5. קלוט מהמשתמש שני מספרים נוספים אשר יגדירו גודל תת המטריצה *הדפס ערכים אילו למסך.* (שים לב בקלט שגודל השורות והעמודות בתת המטריצה צריך להיות קטן מהשורות והעמודות במטריצה הגדולה).
6. חפש בתוך המטריצה הגדולה תת מטריצה (לפי הגודל שנבחר) אשר סכום איבריה הוא המקסימלי . **יש לקחת בחשבון רק מקומות שהמטריצה הקטנה נכנסת במלואה במטריצה הגדולה.**
7. הדפס את מטריצה המספרים הגדולה ואת מטריצת המקסימום.
8. במידה ויש יותר מתת מטריצה אחת עם סכום מקסימאלי הדפס את הראשונה שמצאת.

במשימה 2 ו 3 נטפל בנתונים של תמונה בעלת 2 צבעים בלבד, לבן ואדום. התמונה תיוצג ע"י מטריצה כאשר לבן מיוצג ע"י המספר 0, אדום ע"י המספר 1. כל ערך במטריצת התמונה מתארת פיקסל של תמונה.



<u>משימה 2 - Find Squares:</u>

מלבן מיוצג ע"י 4 מספרים המציינים את ה (x,y) של הנקודה השמאלית עליונה, רוחב ואורך המלבן בהתאמה.

במשימה זו נחפש מלבנים אדומים בתוך התמונה.

ידוע שמלבן אחד יכול להיות צמוד למלבן אחר רק בפינות. אין קשר בין מלבנים בשני קצוות המטריצה.

כתוב פונקציה *printRects* אשר תקבל מטריצה כמערך חד ממדי ותדפיס את כל המלבנים שיש בתמונה

בנוסף הפונקציה תחזיר כמה מלבנים נמצאו. **שים לב: רק** הפונקציה *printRects* יכולה להדפיס את המלבנים

יש לחלק את הפונקציה לפחות לתתי הפונקציות הבאות:

1. פונקציה המוצאת את נקודת ההתחלה של מלבן חדש.
2. פונקציה המחזירה את ממדי המלבן החדש. (רוחב ואורך)

לדוגמא עבור התמונה שבציור יודפסו למסך המלבנים הבאים:

```
 0   0   1   2
19   0   1   7
 2   1   2   4
 6   2   4   3
 5   7   2   1
 7   8   2   1
 9   9   3   1
12  10   2   1
16  12   3   1
19  13   1   1
```

<u>המשימה מבצעת את הפעולות הבאות:</u>

<u>הערה:</u> כל שלב הוא לפחות פונקציה אחת נפרדת

1. הגדר מטריצה עם מלבנים בגודל שתבחר המוגדר ע"י קבועים הדפס ערכים אילו למסך.

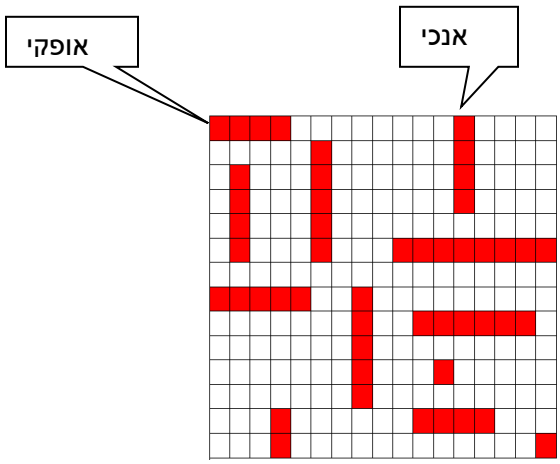
1. הדפס אותה
2. הפעל את הפונקציה *printRects*
3. בפונקציה הראשית הדפס אם נמצאו מלבנים וכמה.

אין להשתמש ב סוגריים [ ] ואין להגדיר עוד מטריצה מלבד התמונה.

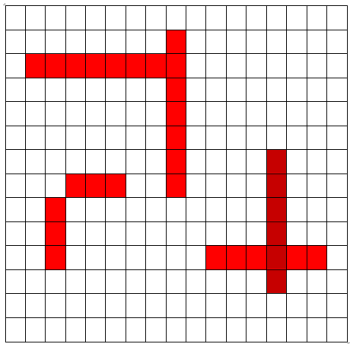
<u>משימה 3 - Add Lines:</u>

בשאלה זו נטפל בקווים בתמונה. קו מיוצג ע"י 4 מספרים המציינים את ה (x,y) של נקודות הקצה של הקו, (x1,y1,x2,y2). במידה והקו אופקי ידוע שנקודה 1 שמאלית ל 2, במידה והקו אנכי ידועה שנקודה 1 עליונה לנקודה 2

- הצורה היחידה האפשרית היא קו אופקי או אנכי.
- כל ה"פיקסלים" בקו צמודים.
- אין חיתוך בין הקווים או הצמדה של קו לקו אחר.
- קו לא "גולש" מהתמונה.



התמונה תקינה



התמונה **לא** תקינה

המשימה מבצעת את הפעולות הבאות:

הערה: כל שלב הוא לפחות פונקציה אחת נפרדת

1. הגדר מטריצה בגודל ROWS שורות ו COLS עמודות בחר ערכים כרצונך. *הדפס ערכים אילו למסך.*
2. קלוט מהמשתמש שני מספרים אשר יגדירו את גודל המטריצה האפקטיבי כך שמספר השורות יהיה בין 1 ל ROWS כולל ומספר העמודות יהיה בין 1 ל COLS כולל. *הדפס ערכים אילו למסך.*
3. צור את המטריצה כך שכל ערכה יהיו 0.

משלב זה אין להשתמש ב סוגריים [ ] ואין להגדיר עוד מטריצה מלבד המטריצה הגדולה.

4. כל עוד המשתמש מעוניין להמשיך ולהכניס קווים לתמונה קלוט 4 מספרים מהמשתמש המייצגים את הקו בסדר: (x1,y1,x2,y2) .
5. בדוק שאכן המספרים מייצגים קו חוקי: נמצא בגבולות התמונה ומייצג קו אופקי או אנכי. אם לא חוקי בקש שנית מהמשתמש נתוני קו.
6. בדוק אם ניתן להכניס את הקו לתמונה לפי החוקיות שהוסברה
7. אם ניתן להכניס את הקו לתמונה הכנס אותו אחרת תן הודעה למשתמש שהקו לא תקין
8. הדפס את התמונה/מטריצה לאחר כל הכנסה

בהצלחה