מכון טכנולוגי חולון
Holon Institute of Technology

COMPUTER SCIENCE DEPARTMENT מחלקה למדעי המחשב

סדנה מתקדמת בתכנות 61108
סמסטר ב' תשפ"ד

מטלה 1

מערכים, מבנים, רשימות מקושרות

שאלה 1:

כתבו פונקציה בשם primeSums המקבלת שני מספרים טבעיים n1 ו- n2 (n2>n1) אשר מחשבת את כל המספרים הנמצאים בטווח של n1 עד n2 שסכום הספרות של כל אחד מהם הוא מספר **ראשוני** (המתחלק רק בעצמו וב- 1).

נגדיר את המבנה הבא לייצוג המספר:

```
typedef struct number
{
        unsigned long long num;
        int sum;
} Number;
```

בו num הוא המספר עצמו ו- sum הוא סכום ספרותיו.

לאחסון התוצאות על הפונקציה לבנות מערך דינאמי של איברים מטיפוס Number כך שעבור שדה num של כל איבר המערך, n1 ≤ num ≤ n2 ו- num עונה לתנאי הנ"ל.
הפונקציה תחזיר את כתובת המערך ותעביר (by reference) את גודלו.
במידה והמערך לא מכיל אף איבר, כתובתו תהיה NULL וגודלו יהיה 0.

דוגמא:

עבור n1=221, n2=233, הפונקציה תיצור את המערך הבא:

num	221	223	227	229	230	232
sum	5	7	11	13	5	7

ותעביר 6.

הבדיקה הכפולה עבור כל מספר האם סכום ספרותיו הוא ראשוני היא לא יעילה. מצד שני הקצאה זמנית של המערך בגודל המקסימלי n2-n1+1 היא לא חסכונית. לכן יש **להכפיל פי 2 את גודל הזיכרון המוקצה** לפי הצורך תוך כדי **מילוי המערך** (ראו את השיטה במצגת 1).

ניתן להניח שיש בזיכרון מספיק מקום להקצאה.

חתימת הפונקציה היא:

```
Number *primeSums(unsigned long long n1, unsigned long long n2, int *p_size);
```

יש להשתמש בפונקציות עזר הבאות:

א. פונקציה int isPrime(int num) אשר מקבלת כפרמטר מספר טבעי num ובודקת האם הוא ראשוני. אם המספר הוא ראשוני, הפונקציה תחזיר 1. אחרת עליה להחזיר 0. עבור num = 1 הפונקציה תחזיר 0. על הפונקציה להיות **יעילה ככל האפשר** ולרוץ בזמן $\sqrt{\text{num}}$.

ב. פונקציה int digitSum(unsigned long long num) אשר מחזירה את סכום הספרות של num.

<u>שאלה 2:</u>

נגדיר את **שכנים של איבר** A[i][j] של מטריצה A כאיברים A[i+1][j], A[i-1][j], A[i][j+1], A[i][j-1]. אם A[i][j] נמצא בשורה\עמודה ראשונה\אחרונה של A אז מספר שכניו יהיה פחות בהתאם.

כתבו פונקציה בשם matrixMaxNeighbor שמקבלת מטריצה A של מספרים שלמים המיושמת כמערך דו-ממדי סטטי וגדליה.
על הפונקציה להקצות מערך דו-ממדי חדש המיישם מטריצה B בעלת אותם גדלים כמו A כך שכל איבר B[i][j] שלה יהיה שווה ל**מקסימום** בין שכני A[i][j].

<u>דוגמא:</u>

$$A: \begin{bmatrix} 5 & 12 & 6 & 8 \\ 4 & 7 & 1 & 9 \\ 13 & 20 & 5 & 2 \\ 18 & 10 & 2 & 6 \end{bmatrix}$$

$$B: \begin{bmatrix} 12 & 7 & 12 & 9 \\ 13 & 20 & 9 & 8 \\ 20 & 13 & 20 & 9 \\ 13 & 20 & 10 & 2 \end{bmatrix}$$

הפונקציה תחזיר את הכתובת של המערך החדש.

ניתן להניח שיש בזיכרון מספיק מקום להקצאה.

חתימת הפונקציה היא:

```
int ** matrixMaxNeighbor( int A[][COLS], int rows, int cols);
```

כך ש- COLS הוא קבוע המוגדר באמצעות #define.

יש להשתמש בפונקציות עזר הבאות:

א. פונקציה int neighborMax(int A[][COLS], int rows, int cols, int i, int j) אשר מקבלת מערך דו-ממדי סטטי עם גדליו וגם קואורדינטות i ו-j של איברו ומחזירה את ערכו השכן המקסימלי של A[i][j].

ב. פונקציה int **allocMatrix (int rows, int cols) אשר מקבלת את גדלי המערך הדו-ממדי, מקצה אותו ומחזירה את כתובתו.

שאלה 3:

כתבו פונקציה בשם createThreeLists המקבלת מטריצה A כמערך דו-ממדי דינאמי וגדליה. הפונקציה מוצאת את שתי הקבוצות הבאות של איברי המטריצה:

א. כל איברי המטריצה שערכם שווים ל**סכום הקואורדינטות** שלהם (i+j). למשל, האיבר A[2][3]=5 שייך לקבוצה כי 2+3=5.

ב. כל איברי המטריצה שערכם מהווים **סדרה חשבונית** ביחד עם ערכי ה-i וה-j שלהם (המתחילה ב-i ומסתיימת ב-A[i][j]). למשל, האיבר A[1][4]=7 שייך לקבוצה כי המספרים 1,4,7 מהווים סדרה חשבונית.

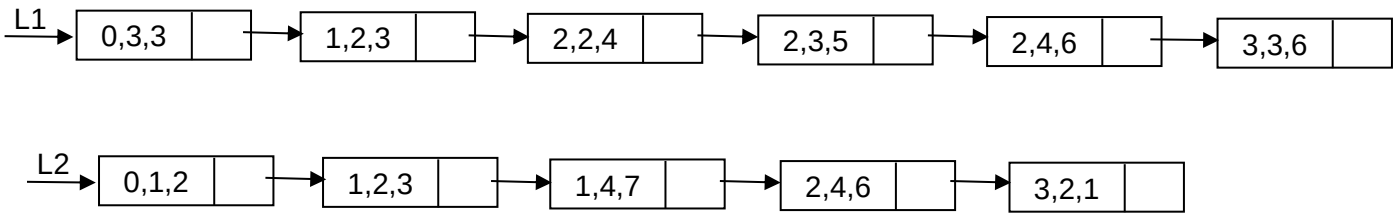
על הפונקציה לבנות שתי רשימות מקושרות L1 ו- L2 כך שאיברים שלהן מכילים שלושה נתונים. כל שלשה תכיל את ערכו של איבר המטריצה שעונה לאחד מהתנאים הנ"ל (תנאי א' – ב- L1, תנאי ב' – ב- L2), והקואורדינטות i ו-j (מס' השורה ומס' העמודה) של אותו האיבר במטריצה.
הפונקציה תעביר (reference by) את שתי הרשימות (ללא איברי דמה בייצוג סופי).

דוגמא:

עבור המטריצה הבאה (איברים של קבוצת א' בלבד – אדומים, איברם של קבוצת ב' בלבד – כחולים, איברים ששייכים לשתי הקבוצות – ירוקים):

i \ j	0	1	2	3	4
0	3	2	5	3	6
1	8	3	3	6	7
2	0	6	4	5	6
3	9	8	1	6	1

הפונקציה יכולה לבנות את שתי הרשימות הבאות (סדר האיברים ברשימה אינו חשוב):



ניתן להניח שיש בזיכרון מספיק מקום להקצאה.

למימוש השלשה יש להשתמש במבנה הבא:

```
typedef struct triad
{
        int i, j, value;
} Triad;
```

ובהתאם לממש את איבר הרשימה באופן הבא:

```
typedef struct item
{
        Triad data;
        struct item *next;
} Item;
```

חתימת הפונקציה היא:

void createThreeLists(int **A, int rows, int cols, Item **pL1, Item **pL2);

יש להשתמש בפונקציות עזר הבאות:

א. פונקציה Triad createThree(int i, int j, int value) המקבלת שלושה מספרים שלמים ומחזירה שלשה אחת המורכבת משלושת הפרמטרים שקיבלה.
ב. פונקציה בשם insert להקצאה והוספת איבר חדש לרשימה. יש לקבוע את חתימתה על פי רצונכם.

הוראות:

1. יש להשתמש כתבנית בקובץ Assignment_1_template.c המצורף אשר מכיל חלק מהקוד. צריך לשתול בתוכו את הטקסטים של כל הפונקציות הנדרשות במקום המתאים.
2. בהתאם לתבנית יש לאחד את כל השאלות הנ"ל לתוכנית אחת באמצעות תפריט הפונקציות Ex1(), Ex2(), Ex3() המשמשות להפעלת פונקציות השאלות 1,2,3 בהתאמה ומטפלות בהכנה ובקליטה של הנתונים המקוריים, בהדפסת התוצאות ובשחרור זיכרון דינאמי.
3. בפונקציה Ex1() יש להצהיר על משתנים, לקלוט את הפרמטרים המקוריים, לקרוא לפונקציה primeSums ולהציג את התוצאות. להדפסת המערך להשתמש בפונקציית עזר void printArray(Number *arr, int size) . יש לשחרר את הזיכרון בסוף.
4. בפונקציה Ex2() יש להצהיר על משתנים (כולל המערך המקורי), לקלוט את המערך באמצעות פונקציית עזר void inputMatrix(int A[ ][COLS], int rows, int cols) ולאימות הקלט להדפיס את המערך הנקלט באמצעות פונקציית עזר void printMatrix(int A[ ][COLS], int rows, int cols) . לאחר מכן לקרוא לפונקציה matrixMaxNeighbor ולהציג את המערך החדש באמצעות פונקציית עזר void printDynamicMatrix(int **A, int rows, int cols) . בסוף יש לשחרר את הזיכרון באמצעות פונקציית עזר void freeMatrix (void **A, int rows) .
5. בפונקציה Ex3() יש להצהיר על משתנים, לקלוט את גדלי המערך הדינאמי המקורי ולהקצות אותו (ניתן להניח שיש בזיכרון מספיק מקום להקצאה) באמצעות פונקציית עזר allocMatrix (שבה משתמשים גם בשאלה 2). לאחר מכן לקלוט את המערך באמצעות פונקציית עזר void inputDynamicMatrix(int **A, int rows, int cols) ולהדפיס את המערך הנקלט באמצעות פונקציית עזר printDynamicMatrix. בשלב הבא צריך לקרוא לפונקציה createThreeLists ולהציג את הרשימות הבנויות באמצעות פונקציית עזר void printList(Item *lst) . בסוף יש לשחרר את הזיכרון באמצעות פונקציות עזר freeMatrix ו- void freeList (Item* lst) .
6. אפשר להשתמש בפונקציות עזר נוספות לפי שיקול דעתכם. ניתן באמצעות #include להוסיף ספריות סטנדרטיות במידת הצורך.
7. אין צורך בבדיקת תקינות הקלט ולא צריך לבדוק בתוך פונקציה את תקינות הפרמטרים שלה.
8. יש להקפיד על ממשק ידידותי ככל האפשר.
9. יש להשתמש בשמות משמעותיים ולתעד את הקוד עם הערות. יש להקפיד לכתוב בצורה מבנית.
10. תכנית שלא עוברת קומפילציה לא תתקבל!