**ביטים**

**שאלה 1**

חיפוש בספר טלפונים.

נניח כי לכל לקוח בספר טלפונים יש 2 מחרוזות:

typedef struct \_client

{

char id[9]; // “12345678”

char phone[12]; // “054-1234567”

} Client;

בשדה ת.ז. ישנן 8 ספרות (ללא ספרת ביקורת) ובסוף ‘\0’.

בשדה טלפון יש שלוש ספרות קידומת, סימן הפרדה ‘-’ (מקף) ואח"כ 7 ספרות המספר עצמו ו – ‘\0’ בסוף.

בסה"כ נשמרים 21 בתים ללקוח.

נרצה להקטין את כמות הבתים הנשמרת ללקוח, מבלי לאבד אינפורמציה.

נשתמש במבנה הבא:

typedef struct \_short\_client

{

unsigned char short\_id[4];

unsigned char short\_phone[5];

} Short\_client;

שימו לב כי כל התווים המשתתפים במידע שנשמר על לקוח הינם עוקבים בטבלת ascii.

מדובר בתווים: ‘0’,’1’,’2’, …’9’, כאשר אין צורך לייצג את המקף כי הוא קבוע ומיקומו ידוע.

ישנם בסה"כ 10 תווים.

ניתן לקודד מחדש כל תו ב – 4 סיביות באופן הבא:

‘0’ 0000

‘1’ 0001

‘2’ 0010

‘3’ 0011

‘4’ 0100

‘5’ 0101

‘6’ 0110

‘7’ 0111

‘8’ 1000

‘9’ 1001

קידוד זה מאפשר לשמור את המידע הרצוי במבנה החדש Short\_client.

א. כתוב פונקציה המקבלת מספר n ומבצעת את הפעולות הבאות:

1. הקצאת מערך דינאמי של n רשומות מסוג Client
2. קליטת נתונים לתוך אברי המערך.
3. דחיסת הנתונים במערך למערך דינאמי של רשומות מסוג Short\_client תוך שחרור המערך הישן.
4. החזרת המערך הדחוס.

ב. כתוב פונקציה המאפשרת חיפוש בספר טלפונים

1. הפונקציה מקבלת מספר ת.ז. כמחרוזת של 9 תווים (8 ספרות ו-'\0').
2. העבר אותו לצורה הדחוסה
3. חפש אותו בלולאה במערך הדחוס. כאשר החיפוש מצליח, על הפונקציה להחזיר את המחרוזת המציינת מהו מספר הטלפון של הלקוח. (המחרוזת הינה בצורה המקורית ולא הדחוסה.)NULL אחרת.

כתוב תוכנית המאפשרת ביצוע הפעולות הנ"ל

**שאלה 2**

א. כתבו את הפונקציה:

int\* filter(int\* Numbers, int size, unsigned char\* pred,

int\* new\_size);

הפונקציה מקבלת מערך Numbersשל מספרים שלמים שונים זה מזה, ואת גודלו size. כמו כן היא מקבלת מערך pred של ביטים, בגודל  בתים. כל ביט במערך pred, מייצג נתון אחד של המערך Numbers, באופן הבא:

הביט ה- 0 בבית ה- 0 של המערך pred, מייצג את הנתון באינדקס 0 במערך Numbers.

הביט ה- 1 בבית ה- 0 של המערך pred, מייצג את הנתון באינדקס 1 במערך Numbers.



הביט ה- 0 בבית ה- 1 של המערך pred, מייצג את הנתון באינדקס 8 במערך Numbers.

וכך הלאה.

הפונקציה תייצר ותחזיר את תת המערך המכיל אך ורק את הנתונים מהמערך Numbers, אשר הביט המתאים להם במערך pred, הוא 1.

כמו כן, הפונקציה תעדכן במשתנה הפלט new\_size את גודלו של תת המערך שייצרה.

למשל, אם המערכים Numbers ו- pred, מוגדרים באופן הבא:

8

4

2

6

30

22

28

32

20

26

24

10

16

14

18

12

Numbers

pred

0 1 0 0 1 0 0 0

0 0 1 0 1 0 0 1

הקריאה filter(Numbers, 16, pred, &new\_size), תייצר את המערך:

וכן תעדכן במשתנה אליו מצביע new\_size את הערך 5.

8

6

22

10

18

הערות:

* הניחו ש- size, מספר הנתונים במערך Numbers, מתחלק ב- 8.
* שימו לב כי בציור, האינדקסים במערך המספרים מתקדמים משמאל לימין, ואילו הביטים בכל בית נספרים מימין לשמאל (מה- LSB אל ה- MSB).

# ב. השתמשו בפונקציה מסעיף א', על מנת לכתוב הפונקציה הבאה:

int\* xorFilter(int\* Numbers, int size, unsigned char\* pred1,  
 unsigned char\* pred2, int\* new\_size);

הפונקציה מקבלת מערך Numbersשל מספרים שלמים שונים זה מזה, ואת גודלו size. כמו כן היא מקבלת שני מערכים pred1 ו- pred2 של ביטים, כל אחד מהם בגודל  בתים. כל אחד מהביטים, בכל אחד מהמערכים pred1 ו- pred2, מייצג נתון מסוים של המערך Numbers, לפי ההתאמה שתוארה בסעיף א'.

הפונקציה תייצר ותחזיר את תת המערך המכיל אך ורק את נתונים מהמערך Numbers. נתון מסוים יופיע במערך התוצאה אם"ם **בדיוק אחד** מביו הביטים המתאימים לו במערך pred1, ו- pred2, הוא 1. כמו כן, הפונקציה תעדכן במשתנה הפלט new\_size את גודלו של תת המערך שייצרה.

הערה: הניחו ש- size, מספר הנתונים במערך Numbers, מתחלק ב- 8.

**מצביעים לפונקציות**

**שאלה 3**

כתבו את הפונקציה:

void\* scramble(void\* arr, int ElemSize, int n, int\* indArr);

הפונקציה מקבלת:

* arr - מערך של איברים מטיפוס כלשהו.
* ElemSize - גודל של כל אלמנט במערך arr
* n - מספר האיברים במערך arr.
* indArr - מערך המכיל תמורה של המספרים: .

על הפונקציה ליצור מערך **חדש** שבו סדר האיברים נקבע על פי המערך indArr באופן הבא:

האיבר ה - i במערך החדש יהיה האיבר שהאינדקס שלו במערך הישן הוא indArr[i].

למשל, אם  ו-  אז:

במקום ה- 0 של המערך החדש ימצא האיבר מהמקום ה- 2 של arr.

במקום ה- 1 של המערך החדש ימצא האיבר מהמקום ה- 0 של arr.

במקום ה- 2 של המערך החדש ימצא האיבר מהמקום ה- 1 של arr.

**שאלה 4**

נתון האלגוריתם הבא למימוש חיפוש בינארי של נתון במערך:

קלט: מערך של נתונים – Arr.

גודל המערך - Size

נתון לחיפוש – Item.

פלט:(true) אם"ם (Item מופיע במערך Arr).

1. found 🡨false

2. left 🡨 0

3. right 🡨 (Size-1)

4. כל עוד ( )



4.1 place 🡨( (left+right) div 2)

4.2 אם (Arr[place]=Item)

4.2.1 found 🡨 true

4.3 אחרת אם (Arr[place]<Item)

4.3.1 left 🡨 (place+1)

4.4 אחרת

4.4.1 right🡨 (place-1)

5. החזר את found

א. כתבו את הפונקציה הבאה:

int binSearch(void\* Arr, int Size, int ElemSize,

void\* Item, int (\*compare)(void\*, void\*));

על הפונקציה לממש אלגוריתם זה , למערכים מטיפוסים שונים (מערכים של מחרוזות, מערכים של int-ים וכו'). הפונקציה מקבלת:

* Arr - כתובת התחלה של מערך.
* Size - מספר הנתונים בו.
* ElemSize - מספר הבתים שתופס כל אחד מהנתונים.
* Item - כתובת של נתון שלגביו יש להכריע האם הוא נמצא במערך. טיפוס הנתון הוא כטיפוס נתוני המערך.
* compare - פויינטר לפונקציה המשווה בין שני נתונים (מהטיפוס של הנתונים מהמערך).

על הפונקציה להחזיר את הערך 1 אם הנתון עליו מצביע Item שווה לאחד מנתוני המערך, או 0 אחרת.

**הערה**:

הניחו שהפונקציה compare, מקבלת כתובות של שני נתונים אותם היא משווה. ומחזירה:

* אפס - אם שני הפרמטרים מצביעים על נתונים שווים.
* מספר שלילי - אם הפרמטר הראשון מצביע על נתון קטן יותר מהנתון שמוצבע ע"י הפרמטר השני.
* מספר חיובי - אם הפרמטר הראשון מצביע על נתון גדול יותר מהנתון שמוצבע ע"י הפרמטר השני.

ב. **השתמשו בפונקציה** BinSearch,שכתבתם, על מנת לכתוב את הפונקציה:

int stringBinSearch(char\*\* strings, int size, char\* str);

המקבלת מערך strings של מחרוזות ממוינות בסדר לקסיקוגרפי, את גודלו size, ומחרוזת

נוספת str.

על הפונקציה להחזיר את הערך 1 אם המחרוזת str מופיעה במערך strings, או 0 אחרת.

למשל אם strings הוא המערך:

“abcdefg”

"good-luck"

"mully"

"stam"

אז ההפעלה: stringBinSearch(strings,4,"good-luck") תחזיר 1.