שאלה 3 (25 נקודות)

שאלה זאת עוסקת במיון מערכים.

סעיף א

בסעיף זה נתון מערך a באורך n, המאוחסן בזיכרון **שהכתיבה אליו מאוד איטית** (בניגוד לקריאה, שהיא מהירה מאוד). במילים אחרות, **בדיקת** התוכן של תא כלשהו במערך הינה פעולה מהירה, אולם **שינוי** ערכו של תא כלשהו הינה פעולה איטית. עליכם לממש פונקציה שתמיין את המערך, תוך שימוש במספר קטן ככל האפשר של כתיבות אליו.

דרישות סיבוכיות: על הפתרון לעבוד ב- O(1) סיבוכיות מקום נוסף (פתרון שלא עומד בדרישה זו לא יתקבל). כאמור, יש לבצע מספר קטן ככל האפשר של **כתיבות** למערך, וכן להשלים את סיבוכיות מספר הכתיבות במקום המתאים למטה. פרט לכך, **אין הגבלה** על סיבוכיות הזמן של הפתרון או על מספר הקריאות מהמערך.

מספר של הכתיבות למערך: $\Theta(n)$

<u>הערה: בסעיף זה ניתן להשתמש בפונקציות שנלמדו בכיתה.</u>

```
void sort_slow(int a[], int n) {
  max_sort(a, n);
}
```

maxsort אומנם עובד בסיבוכיות זמן $\Theta(n^2)$
אבל עושה רק $\Theta(n)$ כתיבות למערך.

סעיף ב

בסעיף זה עליכם שוב למיין מערך של מספרים שלמים. הפעם גודל המערך הוא k+m, כאשר k האיברים הראשונים, במקומות ה-0 עד ה-k-1, מאוחסנים בזיכרון איטי לכתיבה, ואילו m האיברים האחרונים, במקומות ה-k עד ה-k+m-1, מאוחסנים בזיכרון רגיל:

רגיל	רגיל	רגיל	רגיל	איטי	איטי	איטי
k+m-1	**...**	**k+1**	**k**	**k-1**	**...**	**0**

הפונקציה מופיעה בעמוד הבא.

דרישות סיבוכיות:

- עליכם לבצע מספר מועט ככל האפשר של כתיבות לזיכרון האיטי a[0]…a[k-1].
- כמו כן במידת האפשר, על הפתרון גם לעבוד בסיבוכיות זמן טובה ככל האפשר (עם זאת בכל מקרה העדיפות היא למספר כתיבות קטן ככל הניתן ל a[0]…a[k-1]).
- על הפתרון לעמוד בסיבוכיות מקום O(m). פתרון פתרון בסיבוכיות גרועה מזו יקבל ניקוד חלקי בלבד.

השלימו את מדדי הסיבוכיות של הפתרון שלכם במקום המתאים:

מספר הכתיבות לזיכרון האיטי: Θ(k)

סיבוכיות הזמן של הפונקציה: Θ( )

לשאלה זאת כמה פתרונות אפשריים

```
void sort_mixed(int a[], int k, int m) {
  int i;
  for (i=0; i<k; i++){           //this loop puts k smallest
    int min_ind = find_min(a+i,k+m-i); //members in place.
    swap(&a[i], &a[min_ind]);
  }
  mergesort(a+k, m);
}
```

פתרון זה עובד בסיבוכיות זמן $\Theta(k(m+k)+m\log m)$
(לא הכי יעיל, קיבל את רוב הניקוד)

```
void sort_mixed(int a[], int k, int m) {
  int i;
  mergesort(a+k,m);
  for (i=0; i<k; i++) {
    int len = min(k, k+m-i);
    ind min_ind = find_min(a+i,len);
    swap(&a[i], &a[min_ind]);
  }
  mergesort(a+k, m);
}
```

פתרון זה דומה לקודם, אבל קודם ממיינים את החלק השני של המערך, ולכן לולאת חיפוש האיבר הקטן מצתמצמת לגודל k במקום k+m.
פתרון זה עובד בסיבוכיות זמן $\Theta(k^2+m\log m)$

```
void sort_mixed(int a[], int k, int m) {
  int i;
  if (m>=k){
    int* a_copy = (int*)malloc(sizeof(int)*(k+m));
    for (i=0; i<k+m; i++)
      a_copy[i] = a[i];
    mergesort(a_copy, m+k);
    for (i=0; i<k+m; i++)
      a[i] = a_copy[i];
    free(a_copy);
  } else { // k>m
    max_sort(a, k+m);
  }
}
```

פתרון זה בודק מה יותר גדול, m או k.

- אם m יותר גדול אז $O(m)$ מקום לא מגביל אותנו וניצור עותק של מערך, נמיין ונעתיק למערך המקורי. (העותק הכרחי כי mergesort עושה הרבה כתיבות.
- אם k יותר גדול, אז הרי אין מנוס מלעשות k^2 פעולות כדי למיין את החלק האיטי, אז אפשר לעשות זאת גם עבור החלק המהיר (כי במקרה זה הוא אינו גדול מהחלק האיטי)

פתרון זה עובד בסיבוכיות זמן טובה יותר מהפתרון הקודם.