שאלה 2 (25 נקודות)

נתונים שני מערכים: מערך a שאורכו na ומערך b שאורכו nb. איברי שני המערכים הם מטיפוס int. **אנו נניח ש- na≤nb** (כלומר המערך b ארוך יותר) ושכל מערך מכיל איברים שונים זה מזה ללא חזרות.

בשאלה זו (בשני הסעיפים) מותר להשתמש בכל פונקציה או אלגוריתם שנלמדו בכיתה. במידה ובחרתם להשתמש בפונקציות כאלה, ציינו בטבלה הבאה את החתימות שלהן, ואת סיבוכיות הזמן והמקום שלהן.

חתימה	סיבוכיות זמן	סיבוכיות מקום נוסף
void merge_sort(int a[], int n);	Θ(nlogn)	Θ(n)
int binary_search(int a[], int n, int x); **הערה: הגרסה שמחזירה 1 אם האיבר נמצא ו 0 אחרת.**	Θ(logn)	Θ(1)

סעיף א

בסעיף זה נניח כי המערכים **ממוינים**. כתבו פונקציה שמקבלת את שני המערכים, ומחזירה את מספר האיברים שמופיעים רק באחד מן המערכים אך לא בשניהם.

לדוגמה, עבור המערכים הבאים על הפונקציה להחזיר 4, כיוון שיש ארבעה מספרים בדיוק המופיעים רק באחד משני המערכים אך לא בשניהם (המספרים מסומנים בקו):

```
int a[]={1,2,3,5};
int b[]={0,2,3,5,6,7};
```

על הפונקציה לעמוד **בסיבוכיות זמן טובה ככל האפשר**, ובסיבוכיות מקום נוסף O(1). פונקציה לא אופטימלית תזכה בניקוד חלקי בלבד.

שימו לב:

- פתרונות שמגדירים מערכים באורך MAX_INT (או אורך דומה) לא יתקבלו.
- במידת הצורך, ניתן בפונקציה לשנות את תוכן המערכים.

השלימו את סיבוכיות הזמן של הפונקציה שלכם:

סיבוכיות זמן: Θ(_____)

יש 2 פתרונות:

```
int diff(int a[], int b[], int na, int nb) {
   int i=0,j=0,count=0;
   while (i<na && j<nb)
   {
      if (a[i]==b[j])
      {
         i++;
         j++;
      }
      else if (a[i]<b[j])
      {
         i++;
         count++;
      }
      else
      {
         j++;
         count++;
      }
   }
   return count+(na-i)+(nb-j);
}
```

הסבר:
דומה לפעולת merge. עוברים על שני המערכים בו זמנית. אם האיברים זהים – מדלגים עליהם, אחרת סופרים את הקטן וממשיכים.

סיבוכיות זמן: Θ(nb)

```
int diff(int a[], int b[], int na, int nb) {
   int i, count=na+nb;
   for (i=0; i<na; i++)
   {
      if (binary_search(b,nb,a[i]))
         count-=2;
   }
   return count;
}
```

הסבר:
יש סה"כ na+nb איברים בשני המערכים יחד.
עוברים בלולאה על המערך a ומחפשים כל איבר שלו במערך b.

סיבוכיות זמן: Θ(na*log(nb))

הערה:
לא ניתן לקבוע לאיזו משני הפתרונות סיבוכיות טובה יותר, ולכן שני הפתרונות נכונים.

סעיף ב

בסעיף זה **אין להניח** כי המערכים ממוינים. עליכם לממש פונקציה שמקבלת את שני המערכים, ומחזירה את מספר האיברים שמופיעים בשני המערכים בו-זמנית (כלומר, את גודל קבוצת החיתוך). לדוגמה, עבור המערכים הבאים הפונקציה תחזיר את הערך 2 (המספרים שמסומנים בקו משותפים לשני המערכים):

```
int a[]={1,9,5,4};
int b[]={5,10,18,1,3};
```

על הפונקציה להיות בעלת **סיבוכיות זמן טובה ככל האפשר**. בנוסף, במידת האפשר יש לחסוך גם בזיכרון, אך בכל מקרה לא על חשבון סיבוכיות הזמן. פתרונות לא אופטימאליים יזכו בניקוד חלקי בלבד.

שימו לב:

- כיוון ש- nb≥na, אנו נחשיב את θ(na) בתור **טוב יותר** מ- θ(nb) לצרכי השוואת יעילות.
- פתרונות המגדירים מערכים באורך MAX_INT (או אורך דומה) לא יתקבלו.
- במידת הצורך, ניתן בפונקציה לשנות את תוכן המערכים.

השלימו את סיבוכיות הפונקציה שכתבתם:

סיבוכיות זמן: (**nb*log(na)**)Θ סיבוכיות מקום נוסף: (**na**)Θ

```
int intersect(int a[], int b[], int na, int nb) {
   int i, count=0;
   merge_sort(a, na);
   for (i=0; i<nb; i++)
   {
      if (binary_search(a, na, b[i]))
         count++;
   }
   return count;
}
```

הסבר:

ממיינים את המערך a ומחפשים כל איבר של המערך b בתוכו.