מבוא למדעי מחשב מ' / ח' (234114 / 234117)

סמסטר חורף תשס"ז

מבחן מסכם מועד ב', 18 מרץ 2007

מספר סטודנט	*שם משפחה*	*שם פרטי*

משך המבחן: 3 שעות.
חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

הנחיות והוראות:

- מלאו את הפרטים בראש דף זה.
- בדקו שיש 24 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- העמודים הזוגיים בבחינה ריקים. ניתן להשתמש בהם כדפי טיוטה וכן לכתיבת תשובותיכם. סמנו טיוטות באופן ברור על מנת שהן לא תבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר. **ניתן בהחלט להשתמש בעיפרון ומחק**, פרט לדפי השער שאותם יש למלא **בעט בלבד**.
- בכל השאלות ניתן, לבחירתכם, להוסיף הסבר מילולי קצר של הפתרון שלכם (ללא ניקוד). הסבר זה ייקרא ועשוי לעזור לכם במידה והפתרון יהיה בלתי מובן, או יכיל שגיאות רבות.
- בכל השאלות התיכנותיות הינכם רשאים להגדיר ולממש פונקציות עזר כרצונכם. לנוחיותכם, אין חשיבות לסדר מימוש הפונקציות, ובפרט ניתן לממש פונקציה לאחר הפונקציה שמשתמשת בה.
- **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה אלא אם צויין אחרת בשאלה.**

שאלה	ערך	הישג	בודק
1	25		
2	25		
3	25		
4	25		
סה"כ	100		

צוות הקורס
מרצים: סאהר אסמיר, תמיר לוי, דר' מיכאל אלעד (מרצה אחראי).
מתרגלים: איתי שרון, אנדריי קלינגר, אסנת טל, אריה מצליח, ירון יורה, נוגה טל, עידו חניאל, רג'א ג'יריס,רון בגלייטר, סשה סקולוזוב, גיל בכר, צפריר קמלו, רן רובינשטיין (מתרגל אחראי).

בהצלחה!

שאלה 1 (25 נקודות)

סעיף א

נתונה תוכנית ה-C הבאה, כאשר הפרמטר N הוא קבוע #define שלם וחיובי כלשהו:

```
struct pair {
  int a;
  int b;
};


void enigma(int *p1, int *p2)
{
  *p1 *= *p2;
}


void secret(int a, int b)
{
  a--;
  b *= 2;
}


int main ()
{
  int a = N, b = 1;
  struct pair s;
  s.a = a;
  s.b = b;

  while (a > 0) {
    enigma(&b, &a);
    secret(a, s.a);
    secret(b, s.b);
    a--;
  }
  return 0;
}
```

כתבו את ערכם של המשתנים הבאים עם סיום ריצת התוכנית (כלומר כשמגיעים לשורת ה-return). ערכים אלו יש להביע במידת הצורך כפונקציה של N. שימו לב שהכוונה למספר מדויק, ולא לקירוב אסימפטוטי.

a=_____	b=_____	s.a=_____	s.b=_____

סעיף ב

בכל אחד מהסעיפים הבאים מופיעות מספר שורות קוד. לכל קטע קוד, הקיפו בעיגול את התיאור המתאים:

א. **ללא שגיאות** – הקוד יתקמפל ללא כל שגיאה וירוץ ללא תקלות.
ב. **שגיאת זמן ריצה** – הקוד יתקמפל ללא שגיאות, אולם הוא עלול לבצע שגיאה בזמן הריצה.
ג. **שגיאת קומפילציה** – הקוד לא יעבור קומפילציה.

1.
```
int *i = 0;
int  j = *i;
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

2.
```
int a[10] = {0,1,2,3,4,5};
int *p = a;
a = p + 2;
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

3.
```
int *p, *q, a, b, d;
p = &a;
q = &b;
d = (p-q);
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

4.
```
int i=7;
&(i+3) = 10;
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

5.
```
char s[] = "Hello World!";
int i=0;
while (s[i++]);
s[i] = '\0';
```

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

שאלה 2 (25 נקודות)

סעיף א

נתון מערך a המכיל מספרים שלמים וגדולים ממש מאפס, ממוינים בסדר עולה. עליכם לממש פונקציה המקבלת את המערך a, את גודלו n ומספר שלם חיובי ממש x, ומחזירה 1 במידה וניתן להציג את x כמכפלה של שני מספרים מ-a (שימו לב שמכפלה זו יכולה לכלול שני איברים שונים מ-a, או מכפלת איבר בעצמו). במידה ולא ניתן להציג את x כמכפלה של אברים מ-a, הפונקציה תחזיר 0.

לדוגמה, עבור המערך a[6] = {2, 2, 3, 5, 8, 9}, הקריאות two_prod(a,6,18) ו- two_prod(a,6,25) יחזירו שתיהן 1 כיוון ש-2*9=18 וכן 5*5=25. לעומת זאת, הקריאה two_prod(a,6,11) תחזיר 0 כיוון שאין זוג כלשהו של מספרים במערך שמכפלתם 11.

דרישות סיבוכיות: O(n) זמן, O(1) מקום נוסף. פתרון לא אופטימאלי יזכה בניקוד חלקי בלבד.

```
int two_prod(int a[], int n, int x) {
```

סעיף ב

בדומה לסעיף א', נתון מערך a המכיל מספרים שלמים וגדולים ממש מאפס, ממוינים בסדר עולה. עליכם לממש פונקציה המקבלת את המערך a, את גודלו n ומספר שלם כלשהו x, ומחזירה 1 במידה וניתן להציג את x **כהפרש** של שני איברים מ-a. גם כאן, כמו בסעיף א', ניתן בהפרש זה לחסר שני איברים שונים מ-a, או איבר מעצמו. במידה ולא ניתן להציג את x כהפרש של אברים מ-a, הפונקציה תחזיר 0.

לדוגמה, עבור המערך a[6] = {2, 2, 3, 5, 8, 9}, הקריאות two_diff(a,6,7) ו- two_diff(a,6,-3) יחזירו שתיהן 1 כיוון ש-7=2-9 וכן 3- =8-5. לעומת זאת, הקריאה two_diff(a,6,8) תחזיר 0 כיוון שאין זוג כלשהו של מספרים במערך שהפרשם 8.

דרישות סיבוכיות: O(n) זמן, O(1) מקום נוסף. פתרון לא אופטימאלי יזכה בניקוד חלקי בלבד.

```
int two_diff(int a[], int n, int x) {
```

שאלה 3 (25 נקודות)

עבור שני מספרים שלמים a ו-b, כאשר b>a, נגדיר **פרמוטציה 1-חסרה** של התחום [a,b] כמערך המכיל את כל השלמים בין a ל-b, כל מספר בדיוק פעם אחת, פרט לאחד מהם שחסר מהמערך. אנו נאמר שפרמוטציה 1-חסרה היא **ממוינת** אם האיברים בה מסודרים בסדר עולה.

לדוגמה,
1. המערך {4,3,1,7,6,5} הוא פרמוטציה 1-חסרה של התחום [1,7], כיוון שהאיבר 2 חסר.
2. המערך {2,4,5,6} הוא פרמוטציה 1-חסרה ממוינת של התחום [2,6], כיוון שהאיבר 3 חסר.

סעיף א

ממשו את הפונקציה הבאה, המקבלת כקלט מערך p המכיל פרמוטציה 1-חסרה **ממוינת** של התחום [a,b], ומחזירה את הערך החסר במערך. **על מימוש הפונקציה להיות רקורסיבי**; פתרון לא רקורסיבי יזכה לניקוד חלקי בלבד.

דרישות סיבוכיות: O(log(n)) זמן, O(log(n)) מקום נוסף, כאשר n הוא אורך המערך p. שימו לב: פתרון בסיבוכיות גרועה מזו יקבל ניקוד חלקי בלבד.

```
int findsorted(int p[], int a, int b) {
```

<u>סעיף ב</u>

בסעיף זה ניתן להשתמש בפונקצית העזר הבאה:

```
int pivot(int a[], int n, int key);
```

פונקציה זו מקבלת מערך a באורך n וערך key, ומשנה את סדר אברי המערך כך שכל האברים הקטנים או שווים ל-key נמצאים בתחילת המערך, וכל האברים הגדולים ממנו נמצאים בסוף המערך. הפונקציה מחזירה את האינדקס שהחל ממנו ממוקמים האברים הגדולים מ- key במערך. דהיינו, כל האיברים שקטנים או שווים ל-key נמצאים בתאים a[0],...,a[index-1], ואילו האברים הגדולים מ-key נמצאים ב-a[index],...,a[n-1].

הפונקציה pivot() פועלת בסיבוכיות זמן O(n) וסיבוכיות מקום נוסף O(1).

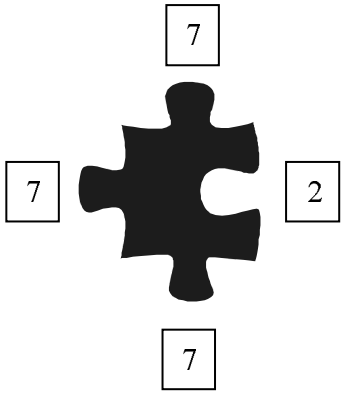
ממשו את הפונקציה הבאה, המקבלת כקלט מערך p המכיל פרמוטציה 1-חסרה **כלשהי** של התחום [a,b], ומחזירה את הערך החסר במערך. **על מימוש הפונקציה להיות רקורסיבי**; פתרון לא רקורסיבי יזכה בניקוד חלקי בלבד. שימו לב ש-p איננו בהכרח ממוין בסעיף זה.

<u>דרישות סיבוכיות</u>: O(n) זמן, O(log(n)) מקום נוסף, כאשר n הוא אורך המערך p. שימו לב: פתרון בסיבוכיות גרועה מזו יקבל ניקוד חלקי בלבד.

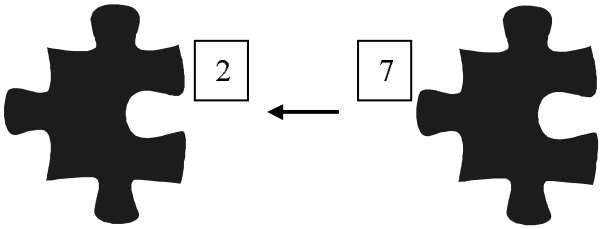
```
int find(int p[], int a, int b) {
```

שאלה 4 (25 נקודות)

בשאלה זו נכתוב תוכנית שמרכיבה פאזל. אנו נניח כי הפאזל מורכב מ- N^2 חתיכות, אותן יש למקם זו לצד זו על פני לוח בגודל NxN. כל חתיכה מהפאזל היא ריבועית ובעלת 4 צלעות, ולכל צלע יש אחת מ-10 צורות חיבור, הממוספרות 0..9. לדוגמה, לחתיכת הפאזל הבאה יש צד ימני עם חיבור מסוג 2, ושלושה צדדים נוספים בעלי חיבור מסוג 7:



על מנת להרכיב את הפאזל, על חתיכות הפאזל השונות להתאים זו לזו בצורת החיבור שלהן. הכלל לחיבור הוא פשוט: חיבור מסוג X מסוגל להתחבר אך ורק לחיבור מסוג (X–9). למשל, עבור החתיכה שלמעלה, הצלע הימנית (חיבור מסוג 2) יכולה להתחבר לצלע השמאלית של חתיכה דומה (חיבור מסוג 7):



אנו נניח, לשם הפשטות, כי כל חתיכות הפאזל מסובבות מראש לכיוון הסופי שלהן בפאזל; כלומר, אין צורך לנסות ולסובב כל חתיכה כאשר מניחים אותה על הלוח. שימו לב שעל מנת למקם חתיכה על הלוח, הצלע הימנית שלה צריכה להתאים לצלע השמאלית של החתיכה שמימינה, הצלע העליונה שלה צריכה להתאים לצלע התחתונה של החתיכה שמעליה, וכן הלאה. עבור חתיכה שנמצאת בקצה הלוח, כל סוג של חיבור יכול להוות שפה של הפאזל (כלומר אין חתיכות מיוחדות שהן פינות או שפות, אלא כל חתיכה יכולה להיות בקצה הלוח).

חלקי הפאזל נתונים במערך דו-ממדי p[N*N][4], כש-N הוא קבוע המוגדר כ-#define. השורה ה-i במערך הדו-מימדי מתאימה לחתיכת הפאזל ה-i, כאשר p[i][0] מכיל את צורת החיבור של הצלע הימנית, p[i][1] את צורת החיבור של הצלע השמאלית, p[i][2] של הצלע העליונה ו-p[i][3] של הצלע התחתונה. לנוחותכם, הוגדרו הקבועים הבאים המייצגים את האינדקסים של 4 הצלעות:

```
#define RIGHT 0
#define LEFT 1
#define TOP 2
#define BOTTOM 3
```

עליכם לממש את הפונקציה solvepuzzle (בדף הבא), המקבלת כקלט את מערך חלקי הפאזל p[N*N][4], ומחזירה 1 במידה ויש פתרון חוקי לפאזל ו-0 אחרת. הפונקציה מקבלת גם מערך פלט דו-ממדי board[N][N] המייצג את הלוח. במידה ויש פתרון, יש לכתוב למערך זה את הסידור של חלקי הפאזל על הלוח – בכל תא יש לכתוב את האינדקס של חתיכת הפאזל שנמצאת בתא זה. במידה ואין פתרון, אין חשיבות לתוכן הלוח בתום הריצה. שימו לב שהמערך board אינו בהכרח מאותחל לתוכן כלשהו בעת הקריאה לפונקציה.

לנוחותכם, ניתן בפתרון להשתמש בפונקצית העזר הבאה. פונקציה זו מקבלת את הלוח board, את מערך החלקים p, אינדקסים i,j של תא בלוח ואינדקס id של אחת מחתיכות הפאזל. הפונקציה מחזירה 1 במידה וניתן למקם את חתיכת הפאזל בתא i,j, ו-0 אחרת. **שימו לב** שפונקציה זו מניחה שתאים ריקים בלוח מכילים את הערך 1-.

```
int islegal(int board[N][N], int p[N*N][4],
            int i, int j, int id)
{
  if (board[i][j] != -1)
    return 0;

  // upper piece
  if (i>0 && board[i-1][j] != -1 &&
      p[board[i-1][j]][BOTTOM] != 9-p[id][TOP])
    return 0;

  // lower piece
  if (i<N-1 && board[i+1][j] != -1 &&
      p[board[i+1][j]][TOP] != 9-p[id][BOTTOM])
    return 0;

  // left piece
  if (j>0 && board[i][j-1] != -1 &&
      p[board[i][j-1]][RIGHT] != 9-p[id][LEFT])
    return 0;

  // right piece
  if (j<N-1 && board[i][j+1] != -1 &&
      p[board[i][j+1]][LEFT] != 9-p[id][RIGHT])
    return 0;

  return 1;
}
```

```
int solvepuzzle(int p[N*N][4], int board[N][N]) {
```