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

סמסטר אביב 2009

מבחן מסכם מועד א', 15 יולי 2009

מספר סטודנט

234114 / 234117

מספר קורס (הקף בעיגול)

משך המבחן: 115 דקות
חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

הנחיות והוראות:

- מלאו את הפרטים בראש דף זה ובדף השער.
- בדקו שיש 16 עמודים (3 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 תיבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר. **ניתן בהחלט להשתמש בעיפרון ומחק (למרות מה שכתוב בדף הראשון!).**
- אין לכתוב הערות והסברים לתשובות אם לא נתבקשתם מפורשות לכך.
- בכל השאלות, הינכם רשאים להגדיר (ולממש) פונקציות עזר כרצונכם.
- **אין להשתמש בפונקציות ספריה, או בפונקציות שמומשו בכיתה אלא אם צוין אחרת במפורש בשאלה, למעט פונקציות קלט פלט והקצאת זיכרון (malloc).**
- בכל שאלה ניתן להשתמש בפונקציות המוגדרות בסעיפים קודמים של אותה שאלה גם אם לא פתרתם סעיפים אלו.

שאלה	ערך	הישג	בודק
1	33		
2	33		
3	34		
סה"כ	100		

צוות הקורס 234114/7

מרצים: פרופ' חבר יובל רבני (מרצה אחראי), מר רן רובינשטיין.
מחליף לאחראי הקורס: פרופ' ראובן בר-יהודה

מתרגלים: זהר קרנין (מתרגל אחראי)
דן רביב, אייל רגב, אופיר ווברר, אייל רוזנברג, רועי אדדי.

בהצלחה!

שאלה 1 (33 נקודות)

חלק א' (18 נקודות)
נתון קטע התוכנית הבא:

```
int  a[] = {2, 3, 4};
int  b[] = {7, 3, 1, 5};
int* p[] = {a, b+1};

printf("%d %d %d\n", *(a+1), b[a[0]], (*p)[2]);

char c = 'a';
char *s1 = &c;
char *s2 = "awsome";
char *arr[] = {"once", "upon", "a", "time"};
printf("%c %s %s\n", (*s1)+1, s2+arr[3][3]-'c', arr[a[0]]);
x
```

מה יהיה פלט ה-printf הראשון? (9 נקודות)

3 1 4

הערה: שימו לב לכך, שפקודת ה-printf הוקדמה לפני ההצהרות על חלק מהמשתנים כדי להדגיש לאילו מהם היא מתייחסת; למעשה עליה להופיע בסוף ההצהרות.

מה יהיה פלט ה-printf השני? (9 נקודות)

b some a

חלק ב' (15 נקודות)

נתונות הפונקציות הבאות:

```
void g(int n);

void h(int n){
     int i;
     if (n < 1) return;
      for (i = 0; i < n; i++)
          printf("?");
      h(n/2);
}

void f(int n){
   int i;
   if (n < 2) return;
   for (i=0; i<n/2; i+=5)
      printf("!");
   g(n/3);
   g(n/3);
}

void g(int n){
   int i;
   for (i = 0; i < n; i++)
      printf("?");
   f(3*n/2);
}
```

מה סיבוכיות הזמן והמקום שתידרש עבור הקריאה **h(n)**? נמקו בקצרה. (5 נקודות)

סיבוכיות זמן: ($\Theta($ n $)$ סיבוכיות מקום נוסף: ($\Theta($ log(n) $)$

זיכרון: כמות הזיכרון המוקצית בגוף הקריאה לפונקציה היא קבועה, ועומק עץ הקריאות הוא $\Theta(\log(n))$.

זמן: מספר הפעולות בקריאה רקורסיבית (עבור $n>0$) הינו $T(n) = T(n/2)+c_1n+c_2$. פיתוח טלסקופי מביא לתשובה לעיל.

מה סיבוכיות הזמן והמקום שתידרש עבור הקריאה **f(n)**? נמקו בקצרה. (10 נקודות)

סיבוכיות זמן: ($\Theta($ n·log(n) $)$ סיבוכיות מקום נוסף: ($\Theta($ log(n) $)$

זיכרון: כמות הזיכרון המוקצית בגוף כל קריאה לפונקציות **f()** ו-**g()** היא קבועה, ועומק עץ הקריאות הוא $\Theta(\log(n))$.

זמן: נוסחאות הנסיגה עבור מספר הפעולות בקריאה לפונקציות **f()** ו-**g()** הן $T_f(n) = 2T_g(n/3)+c_1n+c_2$ ו- $T_g(n) = T_f(3n/2)+c_3n+c_4$ בהתאמה . בהצבה מתקבל $T_f(n)= 2T_f(n/2)+c_5n+c_6$ עבור קבועים c_5, c_6 מתאימים, ופיתוח טלסקופי מביא לתשובה לעיל.

שאלה 2 (33 נקודות)

במערך דו מימדי, זוג אינדקסים (i,j) מהווה גבעה כאשר הערך בתא a[i][j] גדול ממש מהערכים של כל שכניו, כאשר שכניו של תא הם ארבעת התאים הצמודים אליו שמעליו, מתחתיו, מימינו ומשמאלו. במקרה ורק חלק מהשכנים קיימים (כאשר התא i,j נמצא בשולי המערך) יש להתחשב רק בשכנים הקיימים.

למשל עבור המערך הבא בגודל 3X3 :

j →

i ↓	0	1	2
0	26	27	7
1	25	34	8
2	19	29	30

התא (1,1) הוא גבעה, וגם התא (2,2) הוא גבעה כי הוא גדול משני שכניו.

חלק א' (16 נקודות)

כתבו פונקציה (בעמוד הבא) המקבלת כקלט מערך דו מימדי, ומחפשת האם קיימת בו גבעה. הפונקציה תחזיר 1 אם קיימת גבעה ו-0 אחרת. כמן כן, הפונקציה תקבל שני מצביעי פלט ותחזיר בעזרתם את האינדקסים בהם נמצאת אחת הגבעות, במידה ויש. במקרה ולא נמצאה גבעה אין חשיבות לתוכן המוחזר דרך המצביעים.

חתימת הפונקציה:

```
int find_hill(int a[N][N], int *i, int *j)
```

למשל תוכנית אשר עושה שימוש בפונקציה:

```
#define N 3
int main()
{
   int i, j;
   int arr[N][N] = {{26,27,7},{25,34,8},{19,29,30}};
   if (find_hill(arr, &i, &j))
   {
      printf("There is a hill at [%d,%d]", i, j);
   }
}
```

```
int find_hill(int a[N][N], int *i, int *j) {
  int k,l;
  for (k=0;k<N;k++) {
    for (l=0;l<N;l++) {
      if (is_hill(a,k,l)) {
        *i=k;
        *j=l;
        return 1;
      }
    }
  }
  return 0;
}
```

נשתמש בפונקצית העזר `is_hill()`, הבודקת האם תא מסויים במערך הינו גבעה. שימו לב כי סדר הבדיקות בתוך תנאי ה-`if` הינו חשוב למניעת חריגה מגבולות המערך (אם אינדקס של שכן כלשהו חורג מגבולות המערך, לא נשווה את ערך התא `a[i][j]` לערך תא שכן זה).

```
int is_hill(int a[N][N], int i, int j) {
  return ( (i-1 < 0  || a[i][j] > a[i-1][j]) &&
           (j-1 < 0  || a[i][j] > a[i][j-1]) &&
           (i+1 >= N || a[i][j] > a[i+1][j]) &&
           (j+1 >= N || a[i][j] > a[i][j+1]) );
}
```

חלק ב' (17 נקודות)

כעת, נתון כי כל האיברים במערך שונים זה מזה. כתבו פונקציה **רקורסיבית** המקבלת כקלט מערך דו מימדי (שכל איבריו שונים זה מזה) ונקודת התחלה. על הפונקציה להדפיס "מסלול טיפוס" המתחיל בנקודת המוצא ומסתיים בגבעה כלשהי, כאשר בכל צעד במסלול מותר לעבור מנקודה לארבעת שכנותיה (כמוגדר בסעיף א') בלבד.

על הפונקציה להיות רקורסיבית. במידה ויש יותר מגבעה אחת יש למצוא מסלול חוקי מנקודת ההתחלה לאחת הגבעות. כמו כן שימו לב כי מהתנאי שכל איברי המערך שונים זה מזה נובע כי חייבת להיות גבעה אחת לפחות.
על הפונקציה לעבוד בסיבוכיות זמן של $O(N^2)$.

למשל עבור המערך a הבא:

j →
i ↓

3	4	5	6
9	26	27	7
21	28	29	8
20	19	22	30

והקריאה find_path(a,0,0), פלט חוקי לדוגמא הינו (ייתכנו עוד פלטים חוקיים):
(0,0) (1,0) (2,0) (2,1) (2,2)

```
void find_path(int a[N][N], int i, int j) {

  printf("(%d,%d) ", i, j);

  if (i-1 >= 0 && a[i-1][j] > a[i][j])
    find_path(a,i-1,j);

  else if (j-1 >= 0 && a[i][j-1] > a[i][j])
    find_path(a,i,j-1);

  else if (i+1 < N  && a[i+1][j] > a[i][j])
    find_path(a,i+1,j);

  else if (j+1 < N  && a[i][j+1] > a[i][j])
    find_path(a,i,j+1);

  else return;  // current location is a hill
}
```

שאלה 3 (34 נקודות)

נתונה מחרוזת המורכבת מהתווים 'a', 'b' ו- 'c' בלבד. נתון כי התווים במחרוזת ממוינים לפי סדר הא"ב. כמו-כן ידוע כי במחרוזת ישנם יותר תוי 'c' (ממש) מאשר תוי 'a' (שימו לב כי בפרט, יש לפחות תו 'c' אחד).

סעיף א (17 נקודות)

ממשו את הפונקציה הבאה, אשר מקבלת מצביע לתחילת המחרוזת, ומחזירה אינדקס של תו כלשהו שאיננו 'a' (שימו לב שבהכרח יש כזה, כיוון שיש לפחות תו 'c' אחד). שימו לב שהאינדקס של התו הראשון במחרוזת מוגדר להיות 0.

על מימוש הפונקציה להיות בסיבוכיות $O(\log n)$ (כאשר n הוא אורך המחרוזת). מימוש בסיבוכיות גרועה מזו יזוכה בניקוד חלקי לכל היותר. על המימוש להיות בסיבוכיות מקום של $O(1)$.

דוגמה: עבור המחרוזת aaabbbcccc , יוחזר ערך כלשהו בין 3 ל-9 (האינדקס של אחד מתוי ה-b או ה-c).

```
int find_bc(char *s) {

  int k = 0;
  while (s[k] == 'a') {
    k = 2*k + 2;
  }
  return k;
}
```

הסבר: נשים לב שלו היה ידוע לנו אורך המחרוזת, היינו יכולים פשוט להחזיר את האינדקס של התו האחרון בה. משום שהאורך אינו ידוע לנו, אנו נאלצים לבדוק בכל פעם את תוכן התו האחרון שידוע לנו עד כה, ומשם להתקדם על פי תוצאות הבדיקה.

בשלב ראשון אנו יודעים רק על קיום תו אחד במחרוזת, ולכן אנו בודקים את תוכן התא ה-0. עתה, אם מצאנו שהתו באינדקס k הינו 'a', מכאן שכל k+1 התווים הראשונים הם 'a'. מהנתון יש לכן במחרוזת לפחות k+2 תווי 'c', וניתן להסיק שיש בסה"כ לפחות 2k+3 תווים במחרוזת (שהאחרון שבהם באינדקס 2k+2). לפיכך, אנו מעדכנים את האינדקס k לאינדקס האחרון שידוע עליו, וממשיכים באותו האופן.

לצורך ניתוח הסיבוכיות, שימו לב כי עבור מחרוזת באורך n, הלולאה תבצע לכל היותר $\log_2(n)$ איטרציות, שכן בכל איטרציה k גדל לפחות פי 2 ותמיד מתקיים כי k<n שכן איננו חורגים מסיום המערך.

סעיף ב (17 נקודות)

ממשו את הפונקציה הבאה, אשר מקבלת מחרוזת ממוינת בדומה לסעיף א' ומחזירה את האינדקס של <u>ההופעה הראשונה</u> של האות 'b' במחרוזת, או 1- אם האות 'b' לא מופיעה במחרוזת (כרגיל, האינדקס של התו הראשון מוגדר כ-0).
למשל, עבור המחרוזת aaabbbcccc, הפונקציה תחזיר 3, שכן אינדקס מספר 3 הוא הראשון המכיל את התו 'b'. עבור המחרוזת aaccccc הפונקציה תחזיר 1- שכן האות 'b' אינה במחרוזת.

בסעיף זה ניתן לעשות שימוש בפונקציה מסעיף א' גם אם לא פתרתם אותו. כמו כן לצורכי ניתוח סיבוכיות ניתן להניח כי פונקציה זו פועלת בסיבוכיות זמן $O(\log n)$ ומקום $O(1)$, גם אם המימוש שלכם בסיבוכיות גרועה מזו.

על הפתרון להיות בסיבוכיות הזמן **הטובה ביותר** האפשרית, מימוש בסיבוכיות פחות טובה יזוכה בניקוד חלקי לכל היותר. כמו כן יש להשלים את סיבוכיות הזמן והמקום של הפונקציה שכתבתם במקום המתאים.

```
int first_b(char *s) {

  int mid, first = 0;
  int last = find_bc(arr);

  while (last>=first) {
    mid = (last+first)/2;
    if (arr[mid]=='b' && (mid == 0 || arr[mid-1]=='a'))
      return mid;
    else if (arr[mid]=='a')
      first = mid+1;
    else
      last = mid-1;
  }
  return -1;
}
```

סיבוכיות זמן: Θ(log(n)) סיבוכיות מקום נוסף: Θ(1)