מבוא למדעי המחשב מ'/ח' (234114 \ 234117)

מועד א' תשפ"ב

מבחן מסכם , 17.02.2022

מספר סטודנט: | | | | | | | | | | *רשום/ה לקורס:* | 2 | 3 | 4 | 1 | 1 | |

משך המבחן: 3 שעות.

חומר עזר: אין להשתמש בכל חומר עזר.

הנחיות כלליות לכולם:

- בדקו שיש 22 עמודים (4 שאלות) במבחן, כולל עמוד זה.
- בכל השאלות, הנכם רשאים להגדיר ולממש פונקציות עזר כרצונכם. לנוחיותכם, אין חשיבות לסדר מימוש הפונקציות בשאלה, ובפרט ניתן לממש פונקציה לאחר השימוש בה.
- אלא אם כן נאמר אחרת בשאלות, **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה**, למעט פונקציות קלט/פלט והקצאת זיכרון (`malloc, free`). ניתן להשתמש בטיפוס `bool` המוגדר ב-`stdbool.h`
- אין להשתמש במשתנים סטטיים וגלובליים אלא אם נדרשתם לכך מפורשות.
- אפשר להניח שקורא התוכנית ידע לפענח את רעיונותיכם אך ורק מקריאת התוכנית אך בנוסף התוכנית יכולה להכיל תיעוד קל להבנה.
- כשאתם נדרשים לכתוב קוד באילוצי סיבוכיות זמן/מקום נתונים, אם לא תעמדו באילוצים אלה תוכלו לקבל בחזרה מקצת הנקודות אם תחשבו נכון ותציינו את הסיבוכיות שהצלחתם להשיג.
- נוהל "לא יודע": אם תכתבו בצורה ברורה "לא יודע/ת" על שאלה (או סעיף) שבה אתם נדרשים לקודד, תקבלו 20% מהניקוד. דבר זה מומלץ אם אתם יודעים שאתם לא יודעים את התשובה.
- נוסחאות שימושיות:

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}=\Theta(\log\ n) \qquad \log 1+\log 2+\ldots+\log n=\Theta(n\ \log\ n)$$

$$1^k+2^k+3^k+\ldots+n^k=\Theta(n^{k+1}) \qquad 1+k+k^2+k^3+\ldots+k^n=\Theta(k^n),\ k>1$$

הנחיות כלליות לנבחנים במתכונת הרגילה:

- מלאו את הפרטים בראש דף זה ובדף השער המצורף, בעט בלבד.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- אנא סמנו טיוטות באופן ברור על מנת שהן לא תיבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר, **ובעט בלבד**.

הנחיות כלליות לפיילוט:

- יש להגיש 5 קבצים מכווצים ב zip. שמות הקבצים:
hw6q1.c, hw6q2.c, hw6q3.c, hw6q4.c, students.txt
- ההגשה באתר https://webcourse.cs.technion.ac.il/234114/ במשימה ייעודית [מועד א'] בלשונית תרגילי בית.
- שימו לב שקיבלתם את קבצי ה C, ובהם שלד של התוכנית. עליכם רק להשלים את המימוש של הפונקציה הנדרשת. לא ניתן לשנות את שם או סיומת הקובץ. לא ניתן להוסיף קבצים.
- ניתן לעבוד בסביבת עבודה ייעודית לשפת C. שימו לב לדגשים הבאים:
  a. על התוכנית להיות כתובה באופן מסודר ומדורג
  b. יש לתת שמות פונקציות ושמות משתנים משמעותיים
  c. ערכים קבועים יש להגדיר בעזרת define
- בשאלה 1 בחרו את האות המייצגת את הסיבוכיות המתאימה מתוך טבלת הפונקציות להלן:

a. $\Theta(1)$
b. $\Theta(loglog\ n)$
c. $\Theta(logn)$
d. $\Theta(\log^2 n)$
e. $\Theta(\sqrt{n})$
f. $\Theta(\sqrt{n}logn)$
g. $\Theta(n)$
h. $\Theta(nlogn)$
i. $\Theta(nlog^2n)$
j. $\Theta\left(n^{\frac{3}{2}}\right)$
k. $\Theta(n^2)$
l. $\Theta(n^2logn)$
m. $\Theta(n^2\log^2 n)$
n. $\Theta(n^3logn)$
o. $\Theta(n^3)$
p. $\Theta(n^4)$
q. $\Theta\left(2^{\sqrt{n}}\right)$
r. $\Theta\left(2^{\frac{n}{3}}\right)$
s. $\Theta\left(2^{\frac{n}{2}}\right)$
t. $\Theta\left(3^{\frac{n}{3}}\right)$
u. $\Theta\left(3^{\frac{n}{2}}\right)$
v. $\Theta(2^n)$
w. $\Theta(n\cdot 2^n)$
x. $\Theta(3^n)$
y. $\Theta(n^2 3^n)$
z. $\Theta(n^3 3^n)$

השתמשו בקובץ בשם hw6q1.c. הקובץ כמובן מתאים למקרה שבו בכל השאלות בחרתם בתשובה "a". מה שעליכם לעשות הוא לשנות את האות "a" לאות המתאימה לתשובה שלכם בכל סעיף. שימו לב שלכל סעיף יש לבחור גם את האות המתאימה לסיבוכיות הזמן (time complexity) וגם את האות המתאימה לסיבוכיות המקום (space complexity).

צוות הקורס 234114/7

מרצים: פרופ' תומר שלומי (מרצה אחראי), גב' יעל ארז, ד"ר יוסי ויינשטיין, מר' איהאב ותד;
מתרגלים: קטרין חדאד (מתרגלת אראית), מג'ד ח'ורי (מתרגל אחראי), רואי בנימין, מתן ממיסטבלוב, דמיטרי רבינוביץ', אור-אל אדיבי, מרוה מועלם, הראל וקנין, שקד ניסנוב, אדיר רחמים, יואב-מתן פרץ, דניאל ליברמן, צביקה לזר, סתיו רות.

שאלה 1 (25 נקודות):

א. (8 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה `f1` המוגדרת בקטע הקוד הבא, כפונקציה של `n`. אין צורך לפרט את שיקוליכם. חובה לפשט את הביטוי ככל שניתן.

```
void f1(int n)
{
    int x=0;
    for (int i =1; i < n; i*=2)
        for (int j = 0; j < i; j+=2)
            x++;
}
```

סיבוכיות זמן: $\theta($_____n_____$)$ סיבוכיות מקום: $\theta($_____1_____$)$

ב. (8 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה `f2` המוגדרת בקטע הקוד הבא, כפונקציה של `n`. אין צורך לפרט את שיקוליכם. חובה לפשט את הביטוי ככל שניתן.
ניתן להניח שסיבוכיות הזמן של malloc ו free היא $\Theta(1)$

```
int f2(int n)
{
    int x =0;
    for (int i =1; i*i <= n; i++)
        for(int j = i; j*j <= n; j+=i)
            free(malloc(++x));

    free(malloc(n));
    return x;
}
```

סיבוכיות זמן: $\theta($___$\sqrt{n}\log n$_____$)$ סיבוכיות מקום: $\theta($_____n_____$)$

ג. (9 נקודות) חשבו את סיבוכיות הזמן והמקום של הפונקציה `f3` המוגדרת בקטע הקוד הבא, כפונקציה של `n`. אין צורך לפרט את שיקוליכם. חובה לפשט את הביטוי ככל שניתן.

```
int aux(int n)
{
  if (n <=1)
      return 1;

  return 1+aux(n/2);
}

void f3(int n)
{
    int m =2;
    while(m <n)
    {
        aux(m);
        m=m*m;
    }
}
```

סיבוכיות זמן: $\theta($_____ $\log n$ _____$)$ סיבוכיות מקום: $\theta($_____ $\log n$ _____$)$

שאלה 2 (25 נק')

הגדרה: מערך של מספרים שלמים נקרא **מערך ממוין בחלקים** אם הוא נוצר ממערך ממוין בסדר עולה ע"י בחירת רצפים **זרים** שאורכם לפחות 2 איברים והפיכת הסדר שלהם. כלומר, אנו לוקחים מערך ממוין בסדר עולה ובוחרים רצפים **זרים** (אינם חופפים) והופכים את סדרם. נבחין כי אם היינו הופכים את הרצפים הללו שוב היינו מקבלים חזרה את המערך הממוין המקורי.

דוגמא: המערך הבא הינו ממוין בחלקים:

64	55	32	37	41	17	12	7	11	4	2

כי הוא נוצר מהמערך הממוין:

64	55	41	37	32	17	12	11	7	4	2

ע"י הפיכת הסדר של שני החלקים המודגשים.

ממשו את הפונקציה:

```
int findPartiallySorted(int* arr, int n, int x);
```

הפונקציה מקבלת מערך **arr** ממוין בחלקים המכיל מספרים שלמים **שונים**, גודל המערך n ומספר שלם x . על הפונקציה להחזיר את האינדקס של המספר x במידה והוא קיים במערך arr, או 1- אם x איננו נמצא במערך. נתון לכם כי אורך כל רצף שהפך את כיוונו במערך arr הוא **לכל היותר** m, וכי מתקיים ש- 1<m<n . **שימו לב:** הערך של m איננו ידוע לפונקציה וכן מספר הרצפים שהפכו את הסדר שלה איננו ידוע.

דוגמא:

- עבור המערך הממוין בחלקים לעיל ו- x=32 הפונקציה תחזיר את המספר 8 שהוא המיקום של המספר 32 במערך הממוין-בחלקים.
- עבור אותו מערך ו- x=15 הפונקציה תחזיר 1-.

דרישות סיבוכיות:

- סיבוכיות זמן: $\theta(m \log n)$
- סיבוכיות מקום נוסף: $\theta(1)$

אם לדעתכם לא עמדתם בדרישות הסיבוכיות, יש לציין את סיבוכיות הפתרון שלכם.

```
int findPartiallySorted(int* arr, int n, int x)
{
    int low = 0, high = n - 1;
    int mid, temp;

    while(low <= high)
    {
        mid = (low + high) / 2;

        if(arr[mid] == x)
            return mid;

        if(arr[mid] > x)
        {
            temp = mid;
            while(temp < high && arr[temp] > arr[temp + 1]) // linear search - to
the right
            {
                temp++;
                if(arr[temp] == x)
                    return temp;
            }
            high = mid - 1;
        }
        if(arr[mid] < x)
        {
            temp = mid;
            while(temp > low && arr[temp] < arr[temp - 1]) // linear search to the
left
            {
                temp--;
                if(arr[temp] == x)
                    return temp;
            }
            low = mid + 1;
        }
    }

        return -1;
}
```

שאלה 3 (25 נקודות) :

הגדרה : בהינתן מחרוזת str, ותו **כלשהו** c, נגדיר ***את הריבוי של c ב- str*** כמספר המופעים של התו c במחרוזת str.

ממשו את הפונקציה:

```
int findShortestMultiSub(char* s, char* t);
```

הפונקציה מקבלת שתי מחרוזות s ו- t המכילות אותיות אנגליות קטנות בלבד, כך שאורך המחרוזת t **קטן ממש** מאורך המחרוזת s. על הפונקציה למצוא את אורך תת-המחרוזת הקצרה ביותר של s המכילה את כל האותיות המופיעות ב- t כולל **הריבוי שלהם**, כלומר, אם אות כלשהיא c מופיעה בריבוי r_c במחרוזת t, אזי בתת-המחרוזת הקצרה ביותר, האות c חייבת להיות בריבוי גדול או שווה ל-r_c.

דוגמאות:

- עבור "s="jaaabcdefght ו- "t="jafa – הפונקציה תחזיר 9, כי תת-המחרוזת הקצרה ביותר ב- s המכילה את כל האותיות ב- t הינה "**jaa**abcde**f**" .
- עבור "s="aaaacbbbcccca ו- "t="aba – הפונקציה תחזיר 4, כי תת-המחרוזת הקצרה ביותר ב- s המכילה את כל האותיות ב- t הינה "**aa**c**b**" .
- עבור "s="abcaefght ו- "t="ay – הפונקציה תחזיר 1-, כי אין שום תת-מחרוזת ב- s שהיא מכילה את כל האותיות ב- t.

הערות:

- אפשר להניח כי אורך המחרוזת t קטן ממש מאורך המחרוזת s.
- אפשר להניח ששתי המחרוזות s ו- t אינן ריקות.
- במידת הצורך, **מותר** להשתמש בפונקציה strlen() שנלמדה בכיתה מבלי לממש אותה.

דרישות סיבוכיות:

- **סיבוכיות זמן:** $\theta(n)$ כאשר n הוא אורך המחרוזת s
- **סיבוכיות מקום נוסף:** $\theta(1)$

אם לדעתכם לא עמדתם בדרישות הסיבוכיות, יש לציין את סיבוכיות הפתרון שלכם.

```
#define ABC 'Z'-'A'+1
#define A 'a'


int findShortestMultiSub(char* s, char* t)
{
   int len1 = strlen(s);
   int len2 = strlen(t);
   int hash_t[ABC] = { 0 };
   int hash_s[ABC] = { 0 };


   for (int i = 0; i < len2; i++)
      hash_t[t[i]-A]++;


   int start = 0, start_index = -1, min_len = len1;
   int count = 0;
   for (int j = 0; j < len1; j++) {
      hash_s[s[j]-A]++;
      if (hash_s[s[j]-A] <= hash_t[s[j]-A])
         count++;
      if (count == len2) {
         while (hash_s[s[start]-A] > hash_t[s[start]-A] ||
hash_t[s[start]-A] == 0) {
            if (hash_s[s[start]-A] > hash_t[s[start]-A])
               hash_s[s[start]-A]--;
            start++;
         }
         int len_window = j - start + 1;
         if (min_len > len_window) {
            min_len = len_window;
            start_index = start;
         }
      }
   }
   if (start_index == -1) {
      return -1;
   }
   return  min_len;
}
```

שאלה 4 (25 נקודות) :

נתונות N ערים שונות שאותן נסמן $0, \dots, N-1$, ומטריצת כבישים בין הערים roads[N][N] המכילה את הערכים 0,1 בלבד. עבור כל שתי ערים $i \neq j$, קיים כביש המחבר בין העיר i לעיר j אם ורק אם $roads[i][j] = 1$. אפשר להניח שכל הכבישים הם דו-כיווניים, כלומר אם $roads[i][j] = 1$ אזי אפשר ליסוע על הכביש מעיר i לעיר j וכן מ- j ל- i על אותו כביש. לכן לכל i, j מתקיים $roads[i][j] = roads[j][i]$.

מסלול בין הערים i ו- j באורך k הוא סדרה של ערים $a_1, \dots, a_{k-1}, a_k$ כך שמתקיים $a_1 = i, a_k = j$ וכל הערים במסלול הן שונות זו מזו, כלומר, אף עיר אינה חוזרת פעמיים, וכן עבור כל שתי ערים סמוכות במסלול a_t ו- a_{t+1} מתקיים $roads[a_t][a_{t+1}] = 1$.

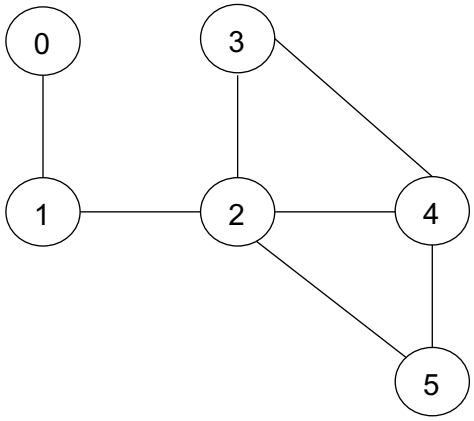
ממשו את הפונקציה:

```
#define N 10

int longest_path(int roads[N][N], int src, int dest);
```

שמקבלת מערך דו ממדי int roads[N][N], ושתי ערים src, dest (אפשר להניח שהם שונות), ומחזירה את אורך המסלול הארוך ביותר בין שתי הערים. אם לא קיים מסלול כזה, על הפונקציה להחזיר 1- . יש להניח כי הקבוע N מוגדר ע"י #define.

לדוגמה, עבור סידור הערים הבא:



המטריצה roads תראה כך:

0	1	0	0	0	0
1	0	1	0	0	0
0	1	0	1	1	1
0	0	1	0	1	0
0	0	1	1	0	1
0	0	1	0	1	0

- עבור הקלט src = 1, dest = 5, הפלט יהיה 5 כלומר, המסלול הנו 1,2,3,4,5 וארכו 5.

המשך השאלה בעמוד הבא...

הערות:

- יש להשתמש בשיטת backtracking כפי שנלמדה בכיתה.
- בשאלה זו אין דרישת סיבוכיות, אולם כמקובל ב- backtracking, יש לוודא שלא מתבצעות קריאות רקורסיביות מיותרות עם פתרונות שאינם חוקיים.
- ניתן ומומלץ להשתמש בפונקציות עזר (ויש לממש את כולן).
- הקבוע N מוגדר ב- #define.

```
#define INVALID -1
#define N 6
int longest_path_aux(int roads[N][N], int src, int dest, bool
*visited);
int longest_path(int roads[N][N], int src, int dest);


int longest_path(int roads[N][N], int src, int dest)
{
   bool visited[N] ={0};
   return  longest_path_aux(roads, src, dest, visited);

}

int longest_path_aux(int roads[N][N], int src, int dest, bool
*visited)
{
   int i = 0, max = -1;
   if(visited[src] == true)
      return INVALID;
   if(src == dest)
      return 1;

   visited[src] = true;
   for(i = 0; i < N; i++)
   {
      if(roads[src][i])
      {
         int k = longest_path_aux(roads, i, dest, visited);
         max = (k+roads[src][i] > max) ? k+roads[src][i] : max;
      }
   }
   visited[src] = false;

   return max;
}
```

בהצלחה!