מדינת ישראל
משרד החינוך

סוג הבחינה: גמר לבתי־ספר לטכנאים ולהנדסאים
מועד הבחינה: אביב תשע"ט, 2019
סמל השאלון: 714911
נספחים: א. דף תשובות
ב. מילון מונחים

מבני נתונים ויעילות אלגוריתמים

הוראות לנבחן

א. משך הבחינה: ארבע שעות.

ב. מבנה השאלון ומפתח ההערכה: בשאלון זה שני פרקים.

פרק ראשון	70	נקודות
פרק שני	30	נקודות
סה"כ	100	נקודות

ג. חומר עזר מותר לשימוש: כל חומר עזר כתוב בכתב־יד או מודפס על נייר.

ד. הוראות מיוחדות:

1. את התשובות לשאלות 1 ו־2 יש לרשום במחברת הבחינה **פרט לתשובתך לשאלה 1 א'**, שאותה יש לענות על־גבי דף התשובות שבנספח א'.

   את התשובות לשאלות 3 ו־4 יש לרשום **אך ורק** על גבי דף התשובות שבנספח א'.

2. לנוחותך, לשאלון זה מצורף מילון מונחים בשפות עברית, ערבית, אנגלית ורוסית. תוכל להיעזר בו בעת הצורך.

הוראות למשגיח:
בתום הבחינה יש לוודא שהנבחנים הדביקו את מדבקת הנבחן שלהם במקום המיועד לכך בדף התשובות שבנספח א' וצירפו אותו למחברת הבחינה.

בשאלון זה 44 עמודים ו־4 עמודי נספחים.

ההנחיות בשאלון זה מנוסחות בלשון זכר,
אך מכוונות הן לנבחנות והן לנבחנים.

בהצלחה!

המשך מעבר לדף

השאלות

<u>פרק ראשון</u> (70 נקודות)

ענה על שתי השאלות 1-2 – שאלות חובה.

שאלה 1 – שאלת חובה (40 נקודות)

בשאלה זו שישה סעיפים (א'-ו'). עליך לענות על <u>כל</u> הסעיפים.

מפעל ייצור קיבל M הזמנות, ולרשותו N ימי עבודה לביצוע.

הנחות יסוד:

1. הביצוע של כל הזמנה דורש יום שלם.
2. לא ניתן לבצע יותר מהזמנה אחת ביום.
3. לכל הזמנה יש תאריך יעד, שהוא המועד האחרון שבו יש לבצע את ההזמנה.
4. לכל הזמנה יש רווח והוא value . הנח כי לכל הזמנה יש value שונה.
5. לא כל M ההזמנות חייבות להתבצע תוך N הימים הנתונים.

מנהלי המפעל מעוניינים לבצע N הזמנות ברציפות, כאשר $M >= N$, כך שהרווח הכולל יהיה מקסימלי, וזקוקים למערכת ממוחשבת שתסייע בשיבוץ ההזמנות לפי הימים בהתאם לאילוצים שפורטו לעיל.

M-N ההזמנות שלא יבוצעו ב־N הימים הנתונים יוכנסו לרשימת ההזמנות שביצוען נדחה.

המשך בעמוד 3

הטבלה שלהלן מציגה את רשימת ההזמנות שנקלטו במפעל הייצור הזה.

לדוגמה:

M = 12 מספר ההזמנות

N = 10 מספר הימים לביצוע

רשימת ההזמנות שנקלטו במפעל

מספר ההזמנה	היום האחרון לביצוע ההזמנה	הרווח (בשקלים)
17	4	1,500
21	3	800
91	10	500
42	5	2,500
62	2	3,000
18	9	600
11	4	200
10	7	3,500
47	8	1,000
12	6	700
33	5	300
49	6	450

המערכת הממוחשבת תתמוך, בין היתר, בפעולות שלהלן:

1. **אתחול המערכת** – **init** – פעולה זו מאתחלת את המערכת, ובין השאר מאפסת את מערך ההזמנות []orders .
   הנח כי פעולה זו מתבצעת פעם אחת בלבד.
2. **הוספת הזמנה למערכת** – **addOrder** – פעולה זו מוסיפה הזמנה חדשה למערכת הממוחשבת.
3. **ביטול הזמנה מן המערכת** – **cancelOrder** – פעולה זו מסירה הזמנה קיימת מתוך המערכת.
4. **שיבוץ** – **schedule** – פעולה זו מטפלת בשיבוץ ההזמנות ל־N ימי העבודה.
5. **מציאת ההזמנה שתתבצע ביום מסוים** – **getOrder** – פעולה זו מציגה את פרטי ההזמנה שתתבצע ביום מסוים.
6. **מציאת היום שבו תתבצע הזמנה מסוימת** – **getDay** – פעולה זו מציגה את היום שבו תתבצע הזמנה מסוימת.
7. **הצגת ההזמנות שביצוען נדחה למועד אחר** – **postponedOrders** – פעולה זו מציגה את ההזמנות שביצוען נדחה למועד אחר (לאחר N הימים הנתונים).

המשך בעמוד 4

לפניך תיאור של מבנה הנתונים התומך במימוש הפעולות הנדרשות מן המערכת הממוחשבת.

נחזיק מבנה (רשומה), שעליו מצביע DS , ואת המבנה נכנה בשם **"המבנה הראשי"** המכיל את השדות האלה:

שדה 1: orders – מערך של הזמנות בגודל MAXIM_ORDERS , כאשר כל תא במערך מכיל פרטי הזמנה כלשהי. מערך זה מייצג טבלת ערבול (טבלת גיבוב – Hash table), ומאופס בתחילה על־ידי הפונקצייה init .

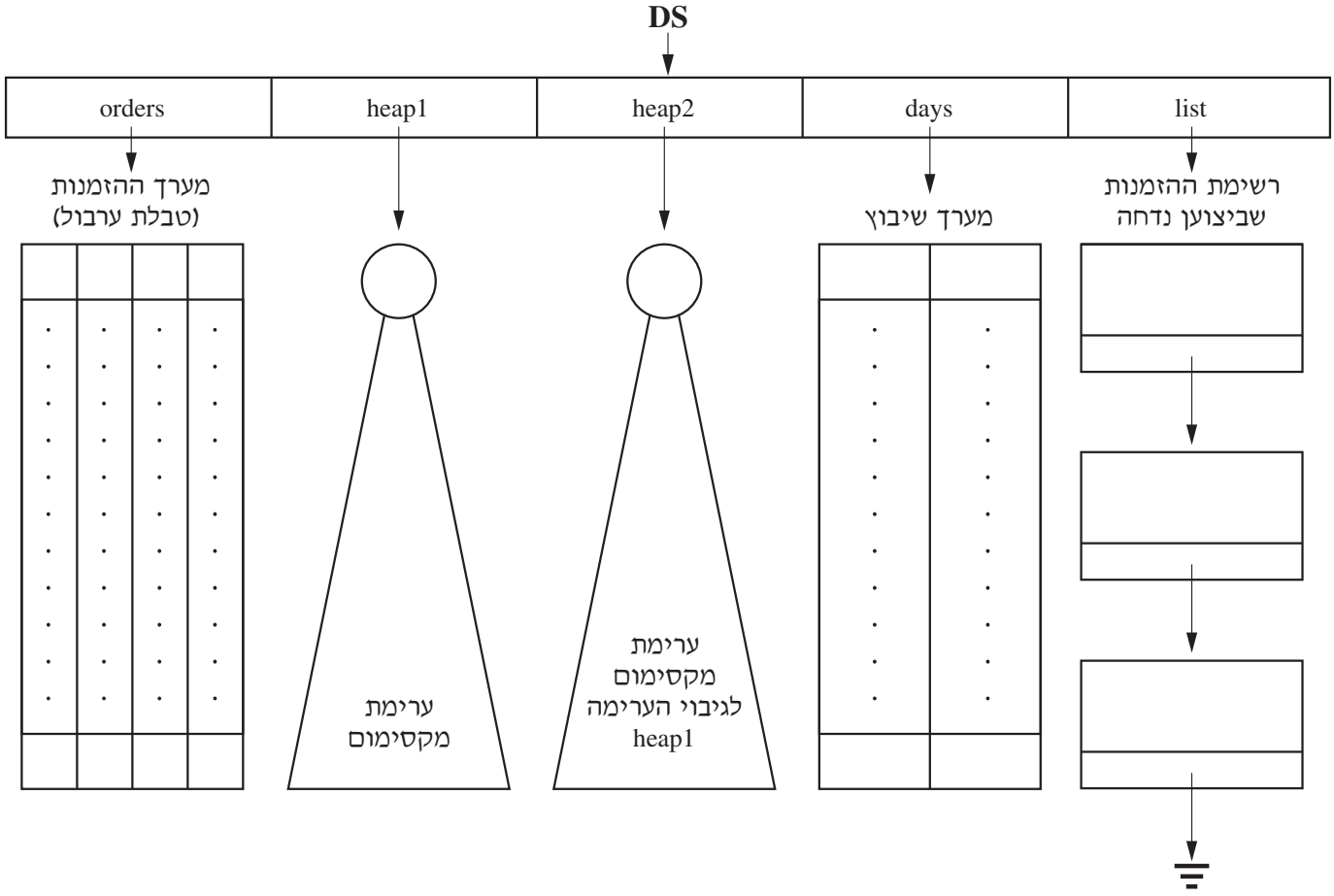
שדה 2: heap1 – מצביע על ערימת מקסימום המסודרת על־פי רווח ההזמנה.

שדה 3: heap2 – מצביע על ערימת מקסימום נוספת המשמשת לגיבוי הערימה heap1 . ערימה זו מסודרת על־פי רווח ההזמנה, ותשמש לעדכון מערך השיבוץ בהמשך.

שדה 4: days – מערך שיבוץ בגודל NUM_OF_DAYS , המציג את שיבוץ ההזמנות לפי ימי העבודה. מספרו הסידורי של איבר במערך זה מייצג את היום שבו תבוצע ההזמנה בפועל, בסדר הכרונולוגי של N ימי העבודה.

שדה 5: list – מצביע לרשימת ההזמנות שביצוען נדחה למועד אחר.

באיור א' מוצג התיאור הסכמתי של "המבנה הראשי", המאגד את כל מבני הנתונים שבהם נאחסן את נתוני המערכת הממוחשבת.



איור א' לשאלה 1

המשך בעמוד 5

להלן הגדרת קבועים בשפת C :

```
#define MAXIM_ORDERS 101

#define NUM_OF_DAYS 10
```

ולהלן הגדרת המבנה הראשי בשפת C :

```
typedef struct headderType          //  טיפוס המבנה הראשי
{
   orderRec orders[MAXIM_ORDERS];   // מערך ההזמנות  (טבלת הערבול)
   heapPtr heap1;                   //  מצביע לערימה
   heapPtr heap2;                   // מצביע לערימת גיבוי
   daysRec days[NUM_OF_DAYS];       // מערך השיבוץ
   listPtr list;                    //  מצביע לרשימת ההזמנות שביצוען נדחה
 } header, *headdPtr;
```

עתה נפרט את מבנה הנתונים בעבור <u>שדה 1</u> של "המבנה הראשי" - מערך ההזמנות.

להלן מבנה של תא **במערך ההזמנות** בשפת C :

```
typedef struct orderType
{
   int orderNum;                    // מספר ההזמנה
   int lastDay ;                    // היום האחרון לביצוע ההזמנה
   int value;                       // רווח ההזמנה
   int dindex;                      // המיקום הסופי של ההזמנה במערך השיבוץ
} orderRec,*orderPtr;
```

המשך בעמוד 6

עתה נפרט את מבנה הנתונים בעבור <u>שדות 2 ו־3</u> של "המבנה הראשי" - ערימות ההזמנות.

להלן מבנה של צומת **בערימת ההזמנות** בשפת C :

```
typedef struct heapType

{
   int value;          // רווח ההזמנה

   int lastDay;        // היום האחרון לביצוע ההזמנה

   int index;          // מיקום ההזמנה בטבלת הערבול

} heapRec,*heapPtr;
```

עתה נפרט את מבנה הנתונים בעבור <u>שדה 4</u> של "המבנה הראשי" - מערך השיבוץ.

להלן מבנה של תא **במערך השיבוץ** בשפת C :

```
typedef struct daysType

{
   int index;          // מיקום ההזמנה בטבלת הערבול

   int value;          // רווח ההזמנה

} daysRec,*daysPtr;
```

עתה נפרט את מבנה הנתונים בעבור <u>שדה 5</u> של "המבנה הראשי" - רשימת ההזמנות שביצוען נדחה.

להלן מבנה של צומת ברשימת **ההזמנות שביצוען נדחה** בשפת C :

```
typedef struct listType

{
   int ID;                     // מספר ההזמנה

   int value;                  // רווח ההזמנה

   struct listType *next;      // מצביע לצומת הבא

} listRec,*listPtr;
```

המשך בעמוד 7

להלן הגדרות התקפות לכל הסעיפים שיבואו בהמשך:

```
typedef enum {FAILURE,SUCCESS,INVALID_INPUT,ORDER_NOT_FOUND,DUPLICATE_ID}
statusType;

typedef enum {FALSE,TRUE} boolean;

typedef enum{GET,PUT} modType;
```

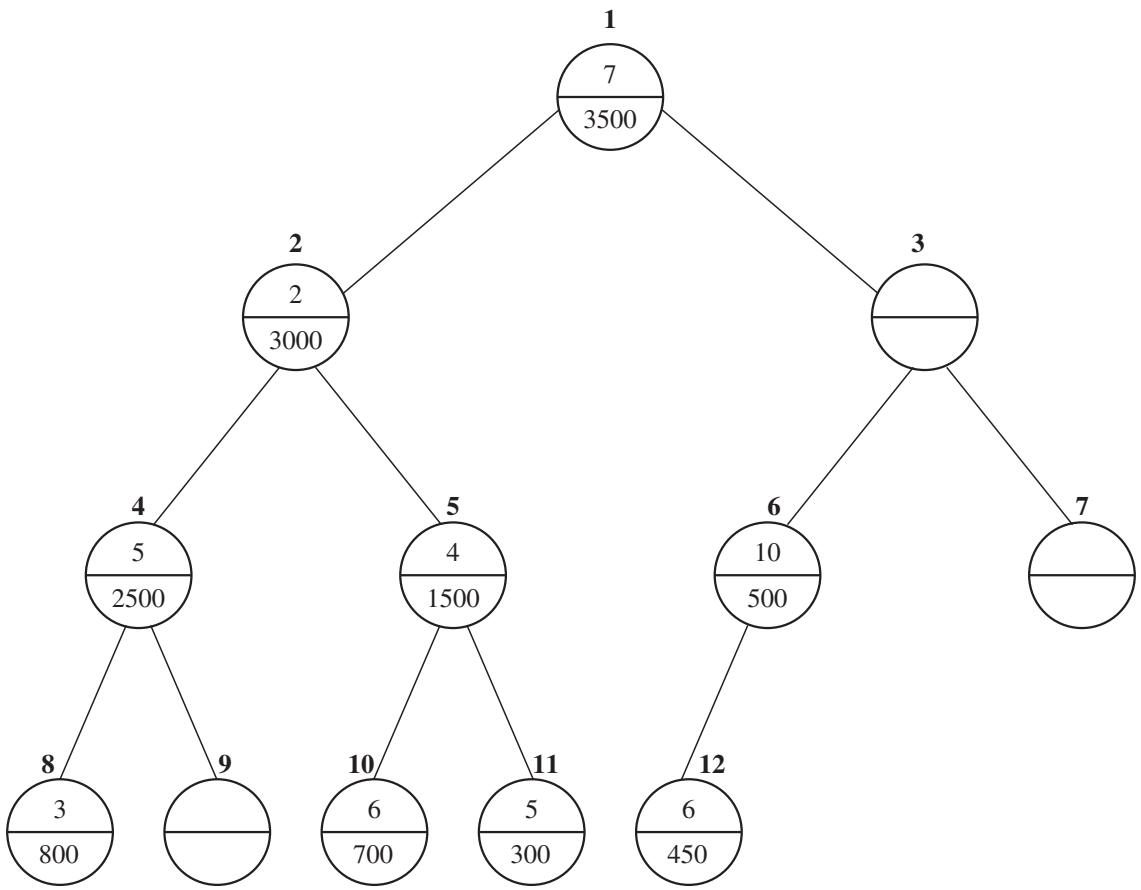
(3 נק') א. להלן אלגוריתם לשיבוץ M ההזמנות:

1. קלוט את M ההזמנות והכנס אותן למערך ההזמנות []orders ולערימה heap1.
2. כל עוד הערימה heap1 לא ריקה, בצע:
   2.1 הוצא הזמנה מתוך הערימה heap1 .
   2.2 הכנס למשתנה־עזר d את היום האחרון לביצוע ההזמנה.
   2.3 אם days[d].value = 0 (היום פנוי) – הכנס את ההזמנה למקום d במערך השיבוץ days[] .
   2.4 אחרת –
   עבור i = d - 1 … 0 , בצע:
   אם days[i].value = 0 (היום פנוי) – הכנס את ההזמנה למקום i במערך השיבוץ days[] , וסיים את הלולאה.
   2.5 אם לא נמצא מקום פנוי – הכנס את ההזמנה לרשימת ההזמנות שביצוען נדחה.
3. הדפס את מערך השיבוץ days[] .

המשך בעמוד 8

הנח כי בטבלה שהובאה בעמוד 3 מוצגת רשימת ההזמנות שנקלטו במערכת.

לאחר בניית הערימה, כאשר ההזמנות מוכנסות על־פי סדר הופעתן בטבלה זו, מתקבלת הערימה שלהלן:



איור ב׳ לשאלה 1

בערימה שלעיל חסרים כמה נתונים.

המספר העליון בכל צומת מציין את היום האחרון לביצוע ההזמנה, והמספר התחתון – את הרווח שלה (מספר ההזמנה לא מצוין).

השלם את הנתונים החסרים בשלושת הצמתים הנותרים **על־גבי איור ב׳ שבנספח א׳.**

רמז: בדיקת מיקומו של צומת שהוכנס לערימה תתחיל מן הסוף.

המשך בעמוד 9

ב. **(4 נק')** סרטט במחברתך את הטבלה שלהלן אשר תכלול את יום ביצוע ההזמנה ואת מספר ההזמנה המתקבלים על־פי האלגוריתם:

יום	מספר ההזמנה
1	.
2	.
3	.
.	.
.	.

ג. **(3 נק')** אילו הזמנות נדחו למועד אחר?

סרטט במחברתך את הטבלה שלהלן אשר תייצג את ההזמנות שביצוען נדחה, המתקבלות על־פי האלגוריתם:

מספר ההזמנה	הרווח
.	.
.	.
.	.
.	.

שים לב: איברי רשימה מקושרת מתווספים לתחילה הרשימה.

המשך בעמוד 10

נתונה ספריית פונקציות, המכילה, בין היתר, את הפונקציות שלהלן:

int **hash**(int num, modType mode)	פונקצייה זו מקבלת שני פרמטרים: num - מספר שלם ו-mode. **אם ערכו של mode הוא PUT** - אז הפונקצייה מחזירה את המיקום בטבלת הערבול אשר אליו יוכנס המספר num . **הערה:** במקרה של הכנסת ערך, הנח כי תמיד יימצא מקום בטבלת הערבול. **אם ערכו של mode הוא GET** - אז הפונקצייה מחפשת בטבלת הערבול את המספר num . אם היא מוצאת אותו, אז היא מחזירה את מיקומו בטבלה זו, אחרת – היא מחזירה את הערך (1-).
void **insertHeap**(heapPtr *ps, heapPtr p, int *capacity, int *size)	פונקצייה זו מקבלת ארבעה פרמטרים: ps – כתובת של מצביע לראש הערימה heap1 , p - מצביע על הצומת שיש להכניס לערימה, capacity – גודל המערך המממש את הערימה, size – הגודל העכשווי של הערימה. הפונקצייה מכניסה לערימה שעליה מצביע ps את הרשימה ש-p מצביע עליה. נוסף על כך, הפונקצייה מעדכנת את גודלה העכשווי של ערימה (size) ובמידת הצורך גם את גודלו של המערך (capacity) .
heapPtr **deleteHeap**(heapPtr *t, int *size)	פונקצייה זו מקבלת שני פרמטרים: t – כתובת של מצביע לראש הערימה, size – הגודל העכשווי של הערימה. הפונקצייה מוציאה צומת מן הערימה ומחזירה מצביע אליו, מסדרת את הערימה מחדש תוך כדי שמירה על תכונות הערימה, ומעדכנת את גודל הערימה size בהתאם.
void **printResults**(daysRec days[])	הפונקצייה מקבלת כפרמטר את days[] , שהוא מערך השיבוץ, ומדפיסה טבלה שמכילה את יום הביצוע של ההזמנה, מספר ההזמנה, הרווח של ההזמנה וכן את סכום הרווח הסופי של כל N ההזמנות שבוצעו.

המשך בעמוד 11

הנח שהפונקציות האלו כתובות, וניתן להשתמש בהן בכל הסעיפים הבאים בלי לכתוב אותן מחדש.
כמו כן, בעבור כל סעיף תוכל להשתמש בכל פונקצייה שמומשה בסעיפים שלפניו.

להלן הגדרות של משתנים גלובליים:

```
headder head;       // המבנה הראשי

headPtr DS = &head;

int heap1Capacity; // הגודל של המערך המממש את הערימה 1 ברגע נתון

int heap1Size;      // כמות האיברים בערימה 1 ברגע נתון

int heap2Capacity; // הגודל של המערך המממש את הערימה 2 ברגע נתון

int heap2Size;      // כמות האיברים בערימה 2 ברגע נתון
```

ענה על הסעיפים שלהלן:

ד. **(7.5 נק')** לפניך פונקצייה שכותרתה:

```
statusType addOrder(int num, int day, int value)
```

פונקצייה זו מקבלת את הפרמטרים האלה:

num – מספר ההזמנה,

day – היום האחרון לביצוע ההזמנה,

value – הרווח של ההזמנה.

פונקצייה זו אמורה להוסיף הזמנה למערכת הממוחשבת (מבצעת הכנסה למערך ההזמנות orders[] ולערימה heap1).

הפונקצייה מחזירה ערך מטיפוס statusType , כמפורט בטבלה שלהלן:

DUPLICATE_ID	אם ההזמנה שמספרה num כבר קיימת בטבלת הערבול
SUCCESS	אם ההזמנה שמספרה num נוספה בהצלחה למערכת הממוחשבת

המשך בעמוד 12

בפונקצייה חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1) – (5) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
statusType addOrder(int num, int day, int value)
 {
  int index;
  heapPtr hp;
  statusType status = SUCCESS;
  if (hash _________(1)_________) status = DUPLICATE_ID;
  else
  {
      index = hash(_________(2)_________);
      DS->orders[index].orderNum = num;
      DS->orders[index].lastDay = day;
      DS->orders[index].value = value;
      hp = _________(3)_________;
      hp->value = value;
      hp->lastDay = day;
      hp->index = index;
      insertHeap(_________(4)_________, hp, _________(5)_________, &heap1Size);
  }
  return status;
}
```

ה. (2 נק') מהי סיבוכיות זמן הריצה של הפונקצייה המוצגת בסעיף ד', אם ידוע ש־n מציין את מספר ההזמנות במערכת הממוחשבת? רשום **במחברתך** את התשובה הנכונה.

1. $O(n)$
2. $O(n \log n)$
3. $O(1)$
4. $O(\log n)$

המשך בעמוד 13

ו. (4.5 נק') לפניך פונקצייה שכותרתה:

```
void insertList(listPtr *list, heapPtr h)
```

פונקצייה זו מקבלת את list , שהוא מצביע לרשימת ההזמנות שביצוען נדחה, ואת h , שהוא מצביע לצומת בודד שהוצא מן הערימה.

הפונקצייה מוסיפה צומת חדש לראש רשימת ההזמנות שביצוען נדחה (רשימה מקושרת).

בפונקצייה חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1) – (3) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void insertList(listPtr *list, heapPtr h)
{
  listPtr l ;
  l = __________(1)__________;
  l->ID = hash(__________(2)__________);
  l->value = h->value;
  l->next = *list;
  __________(3)__________;
}
```

ז. (2 נק') מהי סיבוכיות זמן הריצה של הפונקצייה המוצגת בסעיף ו', אם ידוע ש־n מציין את מספר ההזמנות במערכת הממוחשבת? רשום **במחברתך** את התשובה הנכונה.

1. $O(\log n)$
2. $O(n)$
3. $O(1)$
4. $O(n^2)$

המשך בעמוד 14

(4.5 נק') ח. לפניך פונקצייה שכותרתה:

```
void insertDays(daysRec days[], heapPtr h)
```

פונקצייה זו מקבלת את המערך days[] , שהוא מערך השיבוץ , ואת h , שהוא מצביע לצומת בודד (הזמנה) שהוצא מן הערימה.

הפונקצייה מטפלת בשיבוץ ההזמנה במקום המתאים: במערך השיבוץ days[] או לחילופין ברשימת ההזמנות שביצוען נדחה. כמו כן, הפונקצייה מעדכנת את התא המתאים במערך ההזמנות orders[] (טבלת הערבול).

בפונקצייה חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1) – (3) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
void insertDays(daysRec days[], heapPtr h)
{
  int d = h->lastDay;
  int i,j;
  j = h->index;
  for (i=d; i >= 0;i--)
  {
      if (_____(1)_____)
      {
          days[i].index = h->index;
          days[i].value = h->value;
          DS->_____(2)_____ = i;
          break;
      }
  }
  if (i == -1) _____(3)_____);
}
```

המשך בעמוד 15

ט. **(7.5 נק')** לפניך פונקצייה שכותרתה:

```
void schedule(heapPtr h)
```

פונקצייה זו מקבלת מצביע לראש הערימה heap1 , ומטרתה היא לשבץ את המשימות במערך השיבוץ []days .

לשם כך, הפונקצייה משתמשת בשתי ערימות: heap1 – ערימה ראשית, ו־heap2 – ערימה לגיבוי הערימה heap1 , ומבצעת את הפעולות שלהלן:

1. מאפסת את מערך השיבוץ []days וכן את רשימת ההזמנות שביצוען נדחה.
2. כל עוד הערימה heap1 לא ריקה:
   2.1 מוציאה הזמנה מתוך הערימה heap1 .
   2.2 בודקת אם ההזמנה קיימת **במערך ההזמנות []orders** (אם לא בוטלה).

   אם היא קיימת:

   - בודקת אם יש מקום פנוי **במערך השיבוץ []days** .

     אם יש מקום פנוי – מכניסה את ההזמנה **למערך השיבוץ []days** ,

     אחרת - מעבירה אותה לרשימת ההזמנות שביצוען נדחה.

   - מעבירה את ההזמנה לערימת הגיבוי heap2 .

   הערה: אם ההזמנה לא קיימת במערך ההזמנות, היא לא תוכנס למערך השיבוץ []days וגם לא לערימת הגיבוי (heap2) .
3. בסוף התהליך היא מחליפה את המצביעים של הערימות כך ש־heap1 תצביע על heap2 , ו־heap2 תצביע על heap1 .

בפונקצייה המופיעה בעמוד הבא חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1) – (5) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

המשך בעמוד 16

```
void schedule(heapPtr h)
{
  heapPtr p , temp;
  int c1,s1,i;
  for (i=0;i < NUM_OF_DAYS ;i++)
  {
     DS->days[i].index = 0;
     DS->days[i].value = 0;
  }
  DS->list = NULL;
  do
  {
     p = __________(1)__________;
     if(DS->orders[p->index].orderNum)
      {
        _________(2)__________;
        insertHeap(__________(3)__________, p, &heap2Capacity, &heap2Size);
      }
  } while __________(4)__________;
  c1 = heap1Capacity;
  s1 = heap1Size;
  heap1Capacity = heap2Capacity;
  heap1Size   = heap2Size;
  heap2Capacity = c1;
  heap2Size = s1 ;
  __________(5)__________;
  DS->heap2 = DS->heap1;
  DS->heap1 = temp;
  printResults(DS->days);
}
```

המשך בעמוד 17

(2 נק') **י.** מהי סיבוכיות זמן הריצה של הפונקצייה המוצגת בסעיף ט', אם ידוע ש־n מציין את מספר ההזמנות במערכת הממוחשבת?

הנח כי גודל המערך []days הוא קבוע.

רשום **במחברתך** את התשובה הנכונה.

1. $O(n)$
2. $O(n \log n)$
3. $O(\log n)$
4. $O(\log \log n)$

שאלה 2 – שאלת חובה (30 נקודות)

בשאלה זו שישה סעיפים (א'–ו'). עליך לענות על כל הסעיפים.

כיתת חיילים קיבלה הודעה על יציאה לחופשה, אך נאמר לה שחייל אחד חייב להישאר לשמירה בבסיס.

כדי לבחור את החייל שיישאר בבסיס, הסתדרו החיילים במעגל, והחל מחייל מסוים הם החלו לספור k צעדים. החייל שנמצא במקום ה־k יוצא מן המעגל ורשאי לצאת לחופשה. הספירה מתחילה מחדש החל מהחייל העומד סמוך לחייל שהוצא. וכך הלאה – עד שנשאר חייל יחיד.

החייל הזה יישאר בבסיס.

יש לכתוב תוכנית שתדפיס את שמות החיילים שיוצאים לחופשה בסדר שבו הוצאו מן המעגל, ואת שם החייל שנשאר.

להלן הצעה לפתרון:

יוצרים רשימה מעגלית מקושרת וממוספרת. קובעים מאיזה מספר m במעגל להתחיל את הספירה, ואת מספר הצעדים k שיש לספור. מוציאים מהרשימה את הצומת שנמצא במקום ה־k ומדפיסים את שם החייל המופיע בו. ממשיכים כך עד שנשאר צומת יחיד. צומת זה מייצג את החייל שיישאר בבסיס.

(2 נק') **א.** מהי סיבוכיות זמן הריצה של שיטה זו אם ידוע ש־n מציין את מספר החיילים ו־k מציין את מספר הצעדים?

הנח כי n ו־k אינם קבועים, וכן ש־$k = O(n)$.

רשום **במחברתך** את התשובה הנכונה.

1. $O(n^2)$
2. $O(n)$
3. $O(k \log n)$
4. $O(n \log k)$

המשך בעמוד 18

להלן תוכנית יעילה יותר שתבצע את הנדרש:

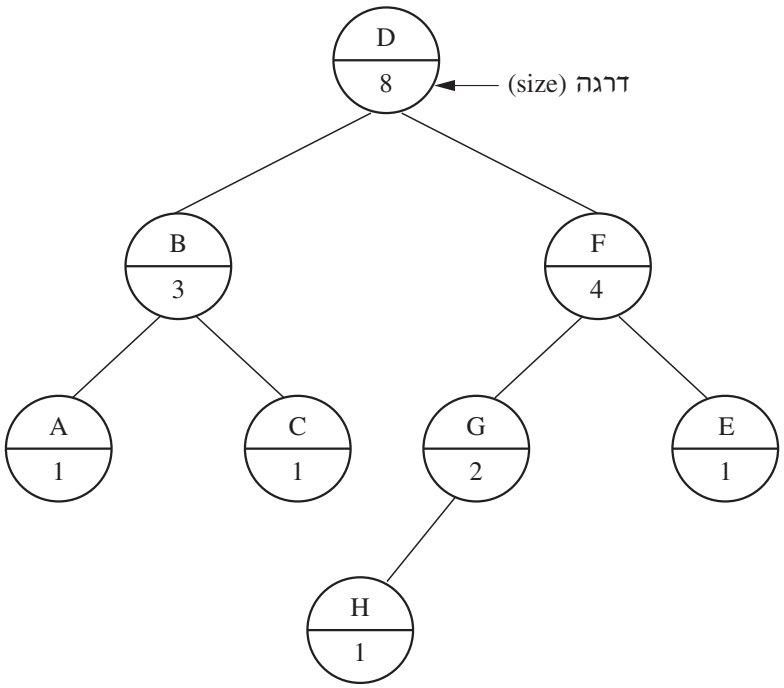
התוכנית בונה מבנה נתונים המכונה "עץ דרגות" (rank tree) בעבור סדרת חיילים מסוימת.

הגדרה: "עץ דרגות" (rank tree) הוא עץ בינרי שבו כל צומת p מכיל, בין היתר, את השדה size , המציין את מספר הצמתים של תת־העץ ששורשו p (כולל p).

בעזרת השדה size ניתן למצוא את החייל ה־k במהירות רבה יותר.

דוגמה: בעבור סדרת החיילים שלהלן: A , B , C , D , E , F , G , H

עץ הדרגות יהיה:



איור א' לשאלה 2

המשך בעמוד 19

להלן הגדרות של המבנים שבהם משתמשת התוכנית:

```
typedef struct elementType  // טיפוס חייל
{
  char Name[20];  // שם החייל
} elementRec,*elementPtr;


typedef struct nodeType  // טיפוס צומת בעץ הדרגות
{
   char Name[20];  // שם החייל
   struct nodeType *left;  // מצביע לתת־העץ השמאלי
   struct nodeType *right;  // מצביע לתת־העץ הימני
   int size;  //    מספר הצמתים של תת־העץ שהצומת הוא שורשו (כולל השורש)
} nodeRec,*nodePtr;
```

להלן הגדרות של משתנים גלובליים:

```
elementPtr elements;
int num, pass, first;
nodePtr iRoot;
```

המשך בעמוד 20

שלבי מהלך התוכנית:

1. **קליטת הנתונים:**

   קליטת מספר החיילים לתוך המשתנה num .

   קליטת שמות החיילים לתוך המערך elements .

   קליטת מספר הצעדים הנספרים לתוך המשתנה pass .

   קליטת המקום שממנו תתחיל הספירה לתוך המשתנה first .

2. **בניית עץ דרגות** מתוך המערך elements בעזרת הפונקצייה buildTree , הבונה עץ דרגות מאוזן.

3. **הדפסת שמות החיילים שיוצאים לחופשה:**

   עבור num > 1 , בצע:

   חשב את מקומו בעץ הדרגות של החייל שיש להוציא מן המעגל.

   הוצא את החייל מעץ הדרגות.

   הדפס את שם החייל שיוצא לחופשה.

   הקטן את ערכו של num.

4. **הדפסת שם החייל שנשאר לשמור בבסיס.**

(8 נק') **ב.** לפניך **שתי** פונקציות:

1. הפונקצייה שכותרתה:

```
void recomputeSize(nodePtr p)
```

פונקצייה זו מקבלת את p שהוא מצביע לצומת בעץ הדרגות.

הפונקצייה מחשבת את ערכו של השדה size בעבור הצומת שעליו מצביע p .

```
void recomputeSize(nodePtr p)
{
  int leftSize, rightSize;
  leftSize = (p->left)? p->left->size:0;
  rightSize = (p->right)? p->right->size:0;
  p->size = leftSize+rightSize + 1;
}
```

המשך בעמוד 21

2. הפונקצייה **הרקורסיבית** שכותרתה:

```
nodePtr buildTree(elementPtr elements,int i,int j)
```

הפונקצייה מקבלת את elements , שהוא מצביע למערך החיילים, וכן את i ו־j , שהם אינדקסים במערך, כאשר: $i \le j$.

הפונקצייה בונה עץ דרגות מאוזן המתאים לסדרת החיילים שמ־elements[i] עד elements[j], ומחשבת בעזרת הפונקצייה recomputeSize את ערכו של השדה size בעבור כל צומת בעץ.

בפונקצייה buildTree חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1) – (4) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
nodePtr buildTree(elementPtr elements,int i,int j)
{
  int mid;
  nodePtr p;
  if(j<i) return NULL;
  if (j==i)
  {
      p = malloc (sizeof(nodeRec));
      strcpy(p->Name,elements[i].Name);
      p->size = 1;
      p->left = NULL;
      p->right = NULL;
      return p;
  }
  else
  {
      mid = (i+j)/2;
      p = malloc (sizeof(nodeRec));
      strcpy(p->Name, _________(1)_________);
      p->left = buildTree(_________(2)_________);
      p->right = buildTree(_________(3)_________);
      _________(4)_________;
  }
  return p;
}
```

המשך בעמוד 22

ג. (6 נק') לפניך פונקצייה **רקורסיבית** שכותרתה:

```
nodePtr findNode(int k,nodePtr t)
```

פונקצייה זו מקבלת את הפרמטרים האלה:

k – מיקום החייל בסדרת החיילים,

t – מצביע לעץ הדרגות.

מטרת הפונקצייה היא למצוא את הצומת של החייל שנמצא במקום ה־k בסדרה בעזרת עץ הדרגות.

אם k שווה ל־size של בנו השמאלי של הצומת t , הפונקצייה מחזירה מצביע לצומת החייל.

אם k קטן מ־size , הפונקצייה פונה שמאלה ומחפשת את k בתת־העץ השמאלי.

אם k גדול מ־size , הפונקצייה פונה ימינה ומחפשת את k–size–1 בתת־העץ הימני.

הפונקצייה מחזירה מצביע לצומת של החייל הנמצא במקום ה־k בסדרה.

בפונקצייה חסרים **שלושה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1) – (3) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
nodePtr findNode(int k,nodePtr t)
{
  int leftSize;
  nodePtr p;
  if(t == NULL) return NULL;
  leftSize = (t->left)? t->left->size:0;
  if(k < leftSize )
  {
     p = _________ (1) _________;
     return p;
  }
  if(k == leftSize) _________ (2) _________;
  p = findNode(_________ (3) _________);
  return p;
}
```

המשך בעמוד 23

ד. **(2 נק')** מהי סיבוכיות זמן הריצה של הפונקצייה findNode אם ידוע ש־n מציין את מספר החיילים?

רשום **במחברתך** את התשובה הנכונה.

1. $O(n)$
2. $O(n^2)$
3. $O(n \log n)$
4. $O(\log n)$

ה. **(8 נק')** לפניך פונקצייה **רקורסיבית** שכותרתה:

```
nodePtr removeNode(int k,nodePtr t)
```

פונקצייה זו מקבלת את הפרמטרים האלה:

k – המספר הסידורי של החייל שיצא לחופשה. הנח כי ערכו של k מחושב בכל איטרציה מחדש בתוכנית הראשית.

t – מצביע לעץ הדרגות.

בעבור החייל ה־k בסדרה, הפונקצייה מוצאת את הצומת שבו משוכן החייל בעץ הדרגות, מדפיסה את שמו ומסירה אותו מן העץ.

הפונקצייה מחשבת מחדש את ה־size של צומתי העץ.

הפונקצייה מחזירה מצביע לראש עץ הדרגות החדש (לאחר הסרת צומת החייל).

הערה: הפונקצייה נעזרת בפונקצייה שכותרתה: nodePtr **removeFirst**(nodePtr t) המקבלת מצביע t לצומת בעץ, מוציאה אותו מתוך העץ, ומחזירה מצביע t לצומת שהוסר.

בפונקצייה חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1) – (4) , בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

המשך בעמוד 24

```
nodePtr removeNode(int k,nodePtr t)
{
  int leftSize;
  if(t==NULL) return NULL;
  leftSize = (t->left)? t->left->size :0;
  if(k < leftSize) t->left = _________(1)_________;
  else if (k > leftSize) t->right= _________(2)_________;
  else
  {
    printf("\%s \t",_________(3)_________);
    t = removeFirst(t);
  }
  if (t) _________(4)_________ ;
  return t;
}
```

המשך בעמוד 25

(4 נק') **ו.** נתונה פונקצייה **רקורסיבית** שאינה קשורה לסעיפים הקודמים:

```
int rec(int n, int k)
{
  if(n==1)
    return n;
  else
    return (rec(n-1, k)+k-1)%n ;
}
```

בצע מעקב אחרי הפונקצייה הזו עבור $n = 5$ ו־$k = 3$ וכתוב מה מחזירה הפונקצייה.

1. 0
2. 1
3. 2
4. 4

המשך בעמוד 26

<u>פרק שני</u> (30 נקודות)

ענה על <u>אחת</u> מבין השאלות 3-4 (לכל שאלה - 30 נקודות).

שאלה 3 (30 נקודות)

בשאלה זו 14 סעיפים. עליך לענות על <u>כל</u> הסעיפים.
בכל סעיף נתונות ארבע תשובות, שרק אחת מהן נכונה. בכל סעיף בחר את התשובה הנכונה, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.

בסעיפים א'-ד' התייחס לבעיה שלהלן:

נתונה סדרה $a_0, a_1 ....a_{n-1}$, של מספרים שלמים. יש למצוא שיטה יעילה למציאת מספרים כפולים בסדרה.

(2 נק') **א.** סורקים לינארית את איברי הסדרה ובודקים לכל איבר a_i אם נמצא בסדרה איבר a_j , $i \neq j$ כך ש: $a_i = a_j$.

מהי סיבוכיות זמן הריצה בשיטה זו?

1. $O(n)$
2. $O(n^2)$
3. $O(n \log n)$
4. $O(1)$

(2 נק') **ב.** מאחסנים את הסדרה במערך $A[i]$, $i = 0, 1, ...n-1$. מבצעים מיון quicksort על המערך.

איברים שווים יימצאו במקומות סמוכים במערך.

מהי סיבוכיות זמן הריצה בשיטה זו?

1. $O(n)$
2. $O(n^2)$
3. $O(n \log n)$
4. $O(1)$

המשך בעמוד 27

(2 נק') **ג.** יוצרים מהסדרה $a_0, a_1 \ldots a_{n-1}$ טבלת ערבול, ובודקים אם $hash(a_i) = hash(a_j)$ עבור $i \neq j$. הנח כי שני מספרים שונים לא יקבלו את אותו מספר ערבול, כלומר אם $a_i \neq a_j$ אז $hash(a_i) \neq hash(a_j)$.

מהי סיבוכיות זמן הריצה בשיטה זו?

1. $O(n)$
2. $O(n^2)$
3. $O(n \log n)$
4. $O(1)$

(2 נק') **ד.** אם ידוע מראש טווח ערכי האיברים שבסדרה, כלומר קיים m קבוע כך ש- $a_i < m$ עבור כל n האיברים של הסדרה – יוצרים מערך VISITED[i] בגודל m כאשר $0 \leq i < m$, ומאחסנים בו את מספר המופעים של כל איבר a_i בסדרה.

מהי סיבוכיות זמן הריצה בשיטה זו?

1. $O(n)$
2. $O(n^2)$
3. $O(n \log n)$
4. $O(1)$

סעיפים ה'-ח' בלתי תלויים זה בזה.

(2 נק') **ה.** נתונה הפונקצייה:

```
int func1(int n)
{
 int i;
 int j;
 int m = 0;
 for(i = 0; i<n; i++)
   for(j = 0; j<sqrt(n); j++)
        m+=1;
 return m;
}
```

המשך בעמוד 28

מהי סיבוכיות זמן הריצה של הפונקצייה הזו?

1. $O(n^{\frac{3}{2}})$
2. $O(n)$
3. $O(\sqrt{n})$
4. $O(n^2)$

(2 נק') **ו.** נתונה הפונקצייה:

```
int func2(int n)
{
  int i;
  int j;
  int m = 0;
  for(i = n; i>0; i = i/2)
      for(j = 0; j<i; j++)
          m+=1;
  return m;
}
```

מהי סיבוכיות זמן הריצה של הפונקצייה הזו?

1. $O(n^2)$
2. $O(n)$
3. $O(n \log n)$
4. $O(\log n)$

המשך בעמוד 29

ז. **(2 נק')** $T(n)=8T\left(\frac{n}{2}\right)+1000\,n^2$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n.

מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(n^2)$
2. $\Theta(n \log n)$
3. $\Theta(n^3)$
4. $\Theta(n^2 \log n)$

ח. **(2 נק')** $T(n)=2T\left(\frac{n}{2}\right)+10\,n$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n.

מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(n^3)$
2. $\Theta(n \log n)$
3. $\Theta(n^2)$
4. $\Theta(n)$

בסעיפים ט'–י"ד התייחס לבעיה שלהלן:

נתון עץ בינארי.

שני צמתים בגרף נקראים **בני־דודים** בהתקיים שני התנאים שלהלן:

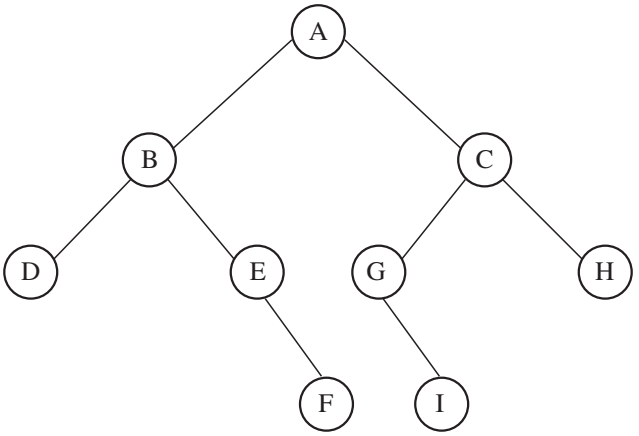
1. שניהם באותה הרמה.
2. הם אינם אחים.

מטרת התוכנית היא לקבל שני צמתים בעץ ולבדוק אם הם בני־דודים (לפי ההגדרה שלעיל).

הערה: בכל סעיף ניתן להשתמש בפונקציות שהוגדרו קודם לכן.

המשך בעמוד 30

(2 נק') **ט.** נתון העץ הזה:



איור א' לשאלה 3

בחר בהיגד הנכון:

1. B ו־C בני־דודים
2. D ו־G בני־דודים
3. E ו־I בני־דודים
4. G ו־H בני־דודים

המשך בעמוד 31

נגדיר צומת של עץ כך:

```
typedef struct nodeType
{
    char ch;
    struct nodeType *left, *right;
} nodeRec, *nodePtr;
```

לפניך פונקצייה **רקורסיבית** שכותרתה:

```
int isSibling(nodePtr root, nodePtr a, nodePtr b)
```

פונקצייה זו מקבלת את root , שהוא מצביע לעץ, ואת a ו־b , שהם שני מצביעים לצמתים בעץ.

הפונקצייה מחזירה 1 אם a ו־b הם אחים (בעלי אב משותף), אחרת - היא מחזירה 0.

בפונקצייה חסרים **שני** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.

```
int isSibling(nodePtr root, nodePtr a, nodePtr b)
{
    if (root==NULL) return 0;
    if ((root->left == a && root->right== b) || (_______(1)________))
         return 1;
    if (isSibling(root->left, a, b))
         return 1;
    return _______(2)________;
}
```

המשך בעמוד 32

(2 נק') **י.** הביטוי החסר (1) הוא:

1. root->left != b || root->right != a
2. root->left == b || root->right == a
3. root->left == b && root->right == a
4. root->left != null && root->right != null

(2 נק') **י"א.** הביטוי החסר (2) הוא:

1. 1
2. 0
3. isSibling (root->left, a, b)
4. isSibling (root->right, a, b)

לפניך הפונקצייה **הרקורסיבית** שכותרתה:

```
int level(nodePtr root, nodePtr ptr, int lev)
```

הפונקצייה מקבלת את root , שהוא מצביע לראש עץ, ואת ptr , שהוא מצביע לצומת בעץ, ואת lev - הרמה ההתחלתית.

הפונקצייה מחזירה את הרמה של הצומת ptr בעץ.

הנח כי הרמה של השורש היא 1.

המשך בעמוד 33

בפונקצייה חסרים **שני** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

```
int level( nodePtr root, nodePtr ptr, int lev)
{
    int t;
    if (root == NULL) return 0;
    if (root == ptr) ________(1)_________;
       t = level(root->left, ptr, lev+1);
    if (t != 0) ________(2)_________;
    return level(root->right, ptr, lev+1);
}
```

(2 נק') **י"ב.** הביטוי החסר (1) הוא:

1. return 1
2. return lev
3. return ptr->data
4. return levt1

(3 נק') **י"ג.** הביטוי החסר (2) הוא:

1. return 1
2. return 0
3. return t
4. return lev

המשך בעמוד 34

לפניך הפונקצייה שכותרתה:

```
int isCousin(nodePtr root, nodePtr a, nodePtr b)
```

הפונקצייה מקבלת את root , שהוא מצביע לשורש העץ, ושני מצביעים a ו־b לצמתים בעץ.

הפונקצייה מחזירה 1 אם a ו־b בני־דודים, אחרת - היא מחזירה 0 .

בפונקצייה חסר ביטוי **אחד**, המסומן במספר בין סוגריים עגולים. בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

```
int isCousin(nodePtr root, nodePtr a, nodePtr b)
{
    if ((level(root,a,1) == level(root,b,1)) && ________(1)_________)
        return 1;
    else
        return 0;
}
```

(3 נק') **י"ד.** הביטוי החסר (1) הוא:

1. isSibling(root, a, b)
2. !isSibling(root, a, b)
3. isCousin(root, a, b)
4. isCousin(root->left, a, b)

המשך בעמוד 35

```
}
```

שאלה 4 (30 נקודות)

בשאלה זו 14 סעיפים. עליך לענות על כל הסעיפים.

בכל סעיף נתונות ארבע תשובות, שרק אחת מהן נכונה.

בכל סעיף בחר את התשובה הנכונה, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.

בסעיפים א'–ז' התייחס לבעיה שלהלן:

יש למצוא את האיבר המינימלי ביותר במחסנית S בסיבוכיות $O(1)$.

להלן הצעה לפתרון: משתמשים בשתי מחסניות: S ו־Min . S תשמש לאחסון רגיל, ו־Min תשמש לאחסון הערכים המינימליים בכל שלב.

נוסף על כך, מגדירים שתי פונקציות נוספות: spush שהיא הרחבה של הפונקצייה push , וכן spop שהיא הרחבה של הפונקצייה pop . כמו כן, יוצרים פונקצייה חדשה getMin המחזירה את הערך המינימלי במחסנית S .

בפעולות שלהלן ניתן להשתמש בפונקציות אלו:

כותרת הפונקצייה	תיאור
void push(stack *S, int x)	פונקצייה זו מקבלת כפרמטר את המחסנית S ומספר נוסף x , ומכניסה אותו למחסנית S .
int pop(stack *S)	פונקצייה זו מוציאה איבר מן המחסנית S ומחזירה אותו כפלט.
boolean isEmpty(stack S)	פונקצייה זו בודקת אם המחסנית S ריקה ומחזירה true או false .

הערה: הנח שהמחסניות S ו־Min הן מחסניות גלובליות.

לפניך הפונקצייה שכותרתה:

```
void spush(int x)
```

פונקצייה זו מקבלת מספר x ומכניסה אותו למחסנית S . במידת הצורך האיבר יוכנס גם למחסנית Min .

המשך בעמוד 36

בפונקצייה חסרים **שני** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.

```
void spush(int x)
{
  int y;
  if (isEmpty(S))
   {
    _______(1)________;
    _______(2)________;
   }
  else
   {
      push(&S,x);
      y = pop(&Min);
      if (x < y)
       {
         push(&Min,y);
         _______(2)________;
       }
      else
         push(&Min, y);
   }
}
```

(2 נק') **א.** הביטוי החסר (1) הוא:

1. push(&Min, y)
2. push(&S, x)
3. push(&S, y)
4. spush(&Min, y)

המשך בעמוד 37

ב. (2 נק') הביטוי החסר (2) הוא:

1. push(&Min, x)
2. spush(&Min, y)
3. push(&Min, y)
4. push(&S, x–y)

לפניך הפונקצייה שכותרתה:

```
int spop()
```

פונקצייה זו מוציאה איבר מהמחסנית S , ובמידת הצורך גם מהמחסנית Min .

בפונקצייה חסרים **שני** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותה בדף התשובות שבנספח א'.

```
int spop()
{
   int x,y;
   x = pop(&S);
   ________(1)_________;
   if (x !=y)
   {
       ________(2)_________;
   }
   return y;
}
```

ג. (2 נק') הביטוי החסר (1) הוא:

1. y = push(&Min, y)
2. y = pop(&S)
3. x = spop(&S)
4. y = pop(&Min)

המשך בעמוד 38

ד. (2 נק') הביטוי החסר (2) הוא:

1. push(&Min, y)
2. push(&S, y)
3. y = pop(&Min)
4. x = pop(&Min)

לפניך הפונקצייה שכותרתה:

```
int getMin()
```

פונקצייה זו מחזירה את הערך המינימלי של המחסנית S ברגע נתון.

בסיום הפונקצייה המחסנית S תישאר ללא שינוי.

בפונקצייה חסרים **שני** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

```
int getMin()
{
  int x;
  ________(1)_________;
  ________(2)_________;
  return x;
}
```

ה. (2 נק') הביטוי החסר (1) הוא:

1. x = pop(&S)
2. x = pop(&Min)
3. push(&S, x)
4. push(&Min, x)

המשך בעמוד 39

(2 נק') **ו.** הביטוי החסר (2) הוא:

1. push(&S, x)
2. x = pop(&S)
3. push(&Min, x)
4. x = pop(&Min)

סעיפים ז'-ט' אינם תלויים זה בזה.

(2 נק') **ז.** נתונה הפונקצייה:

```
int func1(int n)
{
  int i;
  int j=0;
  int m=0;
  for (i=0; i<n; i++)
     for (;j<i; j++)
        m++;
  return m;
}
```

מהי סיבוכיות זמן הריצה של הפונקצייה הזו?

1. $O(n^2)$
2. $O(n)$
3. $O(1)$
4. $O(\sqrt{n})$

המשך בעמוד 40

ח. **(2 נק')** נתונה הפונקצייה:

```
int func2(int n)
{
   int i;
   int j;
   int m=0;
   for (i=1; i<=n; i*=2)
      for (j=1;j<i; j++)
            m++;
   return m;
}
```

מהי סיבוכיות זמן הריצה של הפונקצייה הזו?

1. $O(n)$
2. $O(n^2)$
3. $O(\log n)$
4. $O(n \log n)$

ט. **(2 נק')** $T(n)=2T\left(\frac{n}{2}\right)+n^2$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n.

מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(n^2)$
2. $\Theta(n \log n)$
3. $\Theta(n^3)$
4. $\Theta(n^2 \log n)$

המשך בעמוד 41

בסעיפים י'–י"ד התייחס לבעיה שלהלן:

נתון גרף G בעל V קודקודים ומטריצת צמידות (סמיכויות) A[V][V] .

יש לקבוע אם ניתן לצבוע את קודקודיו בשני צבעים (אדום וכחול) כך שלא יימצאו שני קודקודים סמוכים בעלי אותו הצבע.

להלן אלגוריתם הבודק אם ניתן לצבוע את V קודקודי הגרף G :

1. הכנס קודקוד src לתור Q וצבע אותו באדום.

2. כל זמן שהתור Q לא ריק, בצע:

2.1 הוצא איבר u מ־Q .

2.2 עבור כל קודקוד v הצמוד ל־u בצע:

2.2.1 אם v אינו צבוע, צבע אותו בצבע המשלים לצבעו של u (והכנס אותו לתור Q)

2.2.2 אם הוא צבוע בצבע של u , החזר false וסיים.

3. החזר true .

הנחות:

1. צבעי הקודקודים מאוחסנים במערך colorArr[v] באופן הזה:

$$\text{colorArr}[v]=\left\{\begin{array}{l}\text{אדום } 1\\ \text{כחול } 0\\ \text{ללא צבע } -1\end{array}\right\}$$

2. במצב ההתחלתי כל הקודקודים אינם צבועים.

3. מוגדר מבנה נתונים מטיפוס queue וסדרת פונקציות לטיפול בתור.

להלן פונקצייה שכותרתה:

```
Boolean isBipartite(int G[][V], int  src)
```

פונקצייה זו מממשת את האלגוריתם שהוסבר להלן.

הפונקצייה מקבלת מערך G שמציין את מטריצת הסמיכויות (הצמידות) ואת src - הצומת ההתחלתי.

הפונקצייה מחזירה true אם ניתן לצבוע את הגרף בשני צבעים באופן שצוין לעיל, אחרת - היא מחזירה false .

המשך בעמוד 41

הפונקצייה משתמשת בתור que ובפונקציות־העזר הבאות:

כותרת הפונקצייה	תיאור
void **insertQ**(queue *que, int x)	פונקצייה המכניסה איבר x לתור que
int **deleteQ**(queue *que)	פונקצייה המוציאה איבר מן התור que ומחזירה אותו כפלט
boolean **isEmpty**(queue que)	פונקצייה הבודקת אם התור que ריק

בפונקצייה המופיעה בעמוד הבא חסרים **חמישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספרה המייצגת אותו בדף התשובות שבנספח א'.

המשך בעמוד 43

```
typedef enum {FALSE,TRUE} boolean;

boolean isBipartite(int G[][V], int src)
{
    int colorArr[V];
    int i,v;
    queue  q = NULL;
    for (i = 0; i < V; ++i)
        colorArr[i] = -1;
    colorArr[src] = 1;
    insertQ(_______(1)________);
    while (!isEmpty(q))
    {
        int u = deleteQ(&q);
        if (G[u][u] == 1)
           _______(2)________;
        for ( v = 0; v < V; ++v)
        {
            if (G[u][v] && _______(3)________)
            {
                colorArr[v] = _______(4)________];
                insertQ(&q,v);
            }
            else if (G[u][v] && _______(5)________)
                return FALSE;
        }
    }
     return TRUE;
}
```

המשך בעמוד 44

(2 נק') **י.** הביטוי החסר (1) הוא:

1. q, src
2. q, &src
3. &q, src
4. &q

(2 נק') **י"א.** הביטוי החסר (2) הוא:

1. return TRUE
2. return FALSE
3. return
4. u = delete(&q)

(2 נק') **י"ב.** הביטוי החסר (3) הוא:

1. colorArr[v] != –1
2. colorArr[v] == 0
3. G[u] [v] != –1
4. colorArr[v] == –1

(3 נק') **י"ג.** הביטוי החסר (4) הוא:

1. 1
2. colorArr[u]
3. 1 – colorArr[u]
4. 0

(3 נק') **י"ד.** הביטוי החסר (5) הוא:

1. colorArr[v] == 1
2. colorArr[v] == 0
3. colorArr[v] == colorArr[u]
4. colorArr[v] != colorArr[u]

בהצלחה!

מקום למדבקת נבחן

נספח א': דף תשובות לשאלות 1 א',

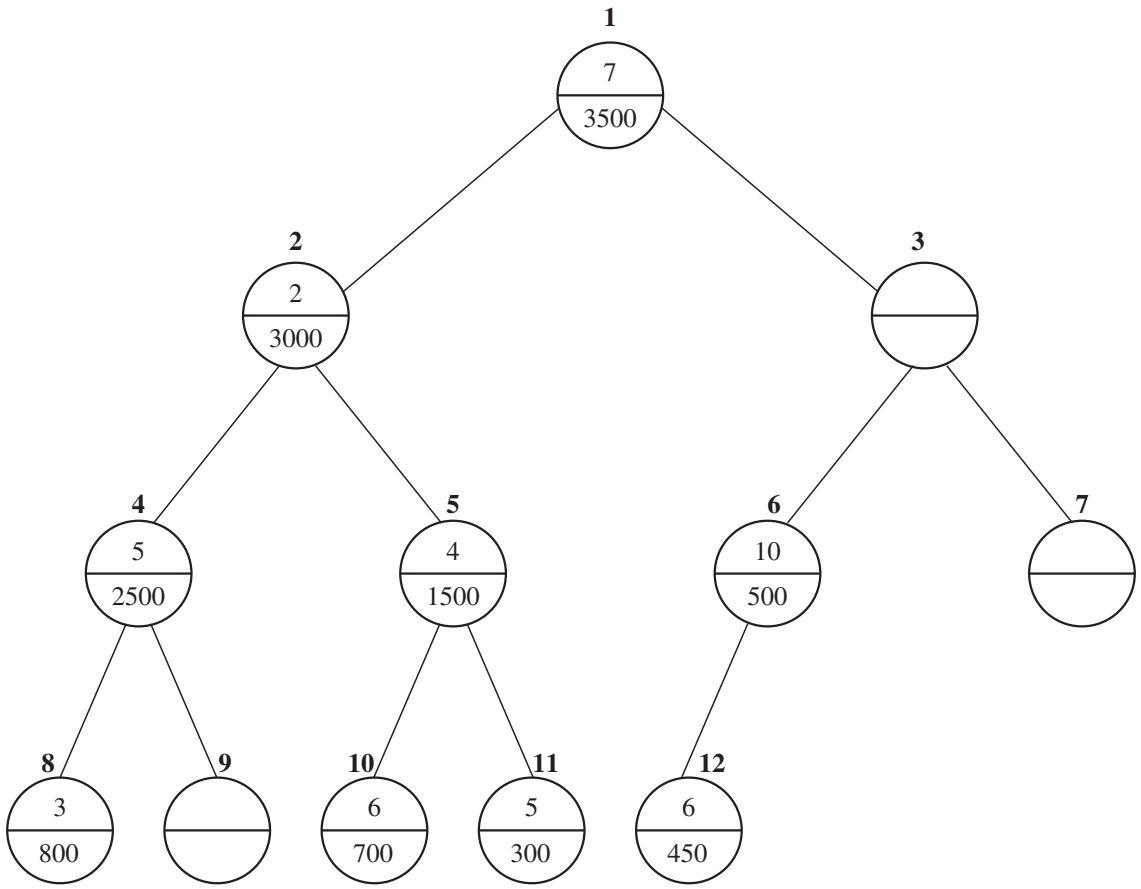
3 ו־4 (2 עמודים)

לשאלון 714911, אביב תשע"ט

הדבק את מדבקת הנבחן שלך במקום המיועד לכך, והדק את הדף הזה למחברת הבחינה שלך.

שאלה 1 סעיף א'

השלם את הנתונים החסרים בשלושת הצמתים הריקים 7,3 ו־9 שבאיור הזה:



איור ב' לשאלה 1

המשך בעמוד 2

הקף בעיגול את הספרה המייצגת את התשובה הנכונה לכל סעיף.

שאלה 3

סעיף א	**1**	**2**	**3**	**4**
סעיף ב	**1**	**2**	**3**	**4**
סעיף ג	**1**	**2**	**3**	**4**
סעיף ד	**1**	**2**	**3**	**4**
סעיף ה	**1**	**2**	**3**	**4**
סעיף ו	**1**	**2**	**3**	**4**
סעיף ז	**1**	**2**	**3**	**4**
סעיף ח	**1**	**2**	**3**	**4**
סעיף ט	**1**	**2**	**3**	**4**
סעיף י	**1**	**2**	**3**	**4**
סעיף י"א	**1**	**2**	**3**	**4**
סעיף י"ב	**1**	**2**	**3**	**4**
סעיף י"ג	**1**	**2**	**3**	**4**
סעיף י"ד	**1**	**2**	**3**	**4**

שאלה 4

סעיף א	**1**	**2**	**3**	**4**
סעיף ב	**1**	**2**	**3**	**4**
סעיף ג	**1**	**2**	**3**	**4**
סעיף ד	**1**	**2**	**3**	**4**
סעיף ה	**1**	**2**	**3**	**4**
סעיף ו	**1**	**2**	**3**	**4**
סעיף ז	**1**	**2**	**3**	**4**
סעיף ח	**1**	**2**	**3**	**4**
סעיף ט	**1**	**2**	**3**	**4**
סעיף י	**1**	**2**	**3**	**4**
סעיף י"א	**1**	**2**	**3**	**4**
סעיף י"ב	**1**	**2**	**3**	**4**
סעיף י"ג	**1**	**2**	**3**	**4**
סעיף י"ד	**1**	**2**	**3**	**4**

נספח ב': מילון מונחים (2 עמודים)

לשאלון 714911, אביב תשע"ט

המונח	תרגום המונח		
	ערבית	רוסית	אנגלית
אחזור	استرجاع	Возврат, извлечение	retrieval
איבר	مُتَغيِّر/عضو	Элемент	item
אציקלי	اسيكليك	Ациклический	acyclic
אקראי	عشوائي	Случайный	random
אתחול	قيمة بدائيّة	Инициализация	initialization
דרגת כניסה/יציאה	درجة الدخول/ الخروج	Степень вершины (входная, выходная)	in-degree, out-degree
זמן ריצה	مدّة التنفيذ	Время работы	run time
חציון	الوسيط	Медиана	median
טבלת ערבול (גיבוב)	جدول الخلط	Хеш-таблица	hash table
טיפוס	نوع	Тип	type
מונוטוני	منبسط	Монотонный	monotonous
מחסנית	باغة	Стек	stack
מטריצת סמיכויות	جدول الحدود الزميلة	Матрица смежности	adjacency matrix
מיון טופולוגי	تصنيف	Топологическая сортировка	topological sorting
מסלול	مسار	Путь	path
מערך דינמי	مصفوفة غير ثابتة	Динамический массив	dynamic array
מצביע	مُؤَشِّر	Указатель	pointer
משתנה גלובלי	مُتَغيِّر عامّ	Глобальная переменная	global variable
סדרה	سلسلة	Последовательность	series
סיבוכיות	تعقيد	Сложность (вычислений)	complexity
עדיפות	أولويّة	Приоритет	preference

המשך בעמוד 2

המונח	תרגום המונח		
	ערבית	רוסית	אנגלית
עץ חיפוש בינארי מאוזן	شجرة بحث ثنائيّ متوازنة	сбалансированное двоичное дерево поиска	balanced binary search tree
עץ פורש	شجرة الامتداد	Остовное дерево	spanning tree
ערך מוחלט	قيمة مُطْلَقة	модуль	absolute value
ערימה	كومة	Куча	heap
ערימה בינארית	كومة ثنائيّة	Двоичная куча	binary heap
פונקציה רקורסיבית	دالّة أو عمليّة تراجعيّة	Рекурсивная функция	recursive function
פונקציית משקל	دالّة لقياس الوزن	Весовая функция	weight function
צומת	مُفْتَرَق	Узел	node
קודקוד	رأس	вершина	vertex
קשת	وصلة	Дуга	arc
רק"ח – רכיב קשיר חזק	مُرَكَّب مرتبط قويّ	компонента сильной связности	strong connected component
רשומה	سِجِلّ	Запись (элемент структуры данных)	record
שדה קישור	حقل رابط	Поле, содержащее ссылку	linking field
שורש	جذر	Корень	root
תת-עץ	شجرة فرعيّة	Поддерево	sub-tree