

סוג הבחינה: גמר לבתי-ספר לטכנאים ולהנדסאים

מועד הבחינה: אביב תשע"ח, 2018

סמל השאלון: 714911

נספחים: א. נספח לשאלה 1

ב. דף תשובות לשאלות 3 ו-4

ג. מילון מונחים

מבני נתונים ויעילות אלגוריתמים

הוראות לנבחן

א. משך הבחינה: ארבע שעות.

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

פרק ראשון	65	נקודות
-----------	----	--------

פרק שני	35	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.

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

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 ב' וצירפו אותו למחברת הבחינה.

בשאלון זה 46 עמודים ו-4 עמודי נספחים.

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

השאלות

פרק ראשון (65 נקודות)

ענה על שתי השאלות 1-2 – שאלות חובה.

שאלה 1 – שאלת חובה (40 נקודות)

בשאלה זו 10 סעיפים (א' – י'). עליך לענות על כל הסעיפים.

לצורך ההכנה לצבא, נפתחה בבית-ספר מסוים תוכנית לשיפור הכושר הגופני של התלמידים. התוכנית כוללת K קבוצות אימון, כאשר K הוא משתנה.

תלמיד שנרשם לתוכנית עובר מבדק כושר גופני, ועל סמך תוצאות המבדק הוא מקבל ציון לרמת הכושר הגופני שלו, שישמש אותו בהמשך.

הנחות יסוד:

1. שיבוץ התלמידים בקבוצות האימון נקבע על-ידי מינהלת התוכנית.
2. מינהלת התוכנית רשאית להעביר תלמיד מקבוצת אימון אחת לאחרת, אולם תלמיד לא יכול להיות משובץ ביותר מקבוצת אימון אחת.
3. מינהלת התוכנית רשאית לאחד שתי קבוצות אימון לקבוצה אחת.
4. תלמיד רשאי לעזוב את התוכנית.
5. מינהלת התוכנית רשאית לבטל קבוצת אימון כלשהי.
6. מספר התלמידים המתקבלים לתוכנית הוא N לכל היותר.

מעוניינים לתכנן מערכת ממוחשבת שתאפשר את ניהול התוכנית לשיפור הכושר הגופני.

המערכת הממוחשבת תתמוך, בין היתר, בפעולות שלהלן:

1. **אתחול המערכת** – `init` – פעולה זו מאתחלת את המערכת הממוחשבת.
הנח כי פעולה זו מתבצעת פעם אחת בלבד.
2. **הוספת תלמיד חדש למערכת** – `addScholar` – פעולה זו מוסיפה תלמיד חדש לקבוצה מסוימת במערכת הממוחשבת.
3. **הוספת קבוצת אימון חדשה למערכת** – `addGroup` – פעולה זו מוסיפה קבוצת אימון חדשה, ריקה, למערכת הממוחשבת.
4. **הסרת תלמיד מן המערכת** – `removeScholar` – פעולה זו מסירה תלמיד קיים מן המערכת הממוחשבת.
5. **מחיקת קבוצת אימון מן המערכת** – `deleteGroup` – פעולה זו מוחקת קבוצת אימון קיימת מן המערכת הממוחשבת. כל התלמידים שהיו רשומים בקבוצה הזו מועברים לקבוצה אחרת במערכת הממוחשבת.
6. **עדכון ציון הכושר הגופני של תלמיד** – `riseScholarLevel` – פעולה זו מעדכנת את ציון הכושר הגופני של תלמיד.
7. **העברת תלמיד מקבוצת אימון אחת לקבוצת אימון אחרת** – `transferScholar` – פעולה זו מעבירה תלמיד מקבוצת אימון אחת לקבוצת אימון אחרת.
8. **מיזוג שתי קבוצות אימון לקבוצה אחת** – `mergeGroup` – פעולה זו מאחדת שתי קבוצות אימון קיימות לקבוצה אחת.
9. **הצגת פרטי קבוצה במערכת** – `printGroup` – פעולה זו מציגה את פרטי הקבוצה. נוסף על כך, היא מציגה את רשימת התלמידים המשתייכים לקבוצה זו. ברשימה זו, בעבור כל תלמיד יוצג שם התלמיד וציון הכושר הגופני שלו. הרשימה תוצג בצורה ממוינת, בסדר עולה, על פי ציון הכושר הגופני של התלמיד.

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: numOfScolars – מספר התלמידים שנרשמו לתוכנית לשיפור הכושר הגופני.

שדה 2: scholarsArr – מערך תלמידים בגודל N, כאשר כל תא במערך זה מכיל פרטי תלמיד
כלשהו. מערך זה מייצג טבלת ערבול (טבלת גיבוב) – Hash table.

שדה 3: groupsArr – מערך דינמי של קבוצות אימון –

כל תא במערך זה מייצג קבוצה אימון אחת, ומכיל, בין היתר, מצביע לשורשו של עץ
מאוזן (עץ AVL) המכיל פרטים מסוימים על התלמידים ששובצו לקבוצת אימון זו.

עץ זה מכונה "עץ כושר", וכאמור – לכל קבוצה יש עץ כושר משלה.

מפתח החיפוש בעץ הכושר הוא ציון הכושר הגופני של התלמיד.

הנחה: ציוני הכושר הגופני של התלמידים שמשובצים באותה קבוצת אימון בהכרח
שונים זה מזה.

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

להלן הגדרת קבוע בשפת C:

```
#define N 100
```

ולהלן הגדרת המבנה הראשי בשפת C:

```
typedef struct headType // טיפוס המבנה הראשי
{
    int numOfScolars; // מספר התלמידים שנרשמו לתוכנית לשיפור הכושר הגופני
    scholarRec scholarsArr[N]; // מערך תלמידים
    groupPtr groupsArr; // מערך דינמי של קבוצות אימון
}header, *headPtr;
```

ענה נפרט את מבנה הנתונים בעבור שדה 2 של "המבנה הראשי" – מערך תלמידים.

להלן מבנה של תא במערך התלמידים בשפת C :

```
typedef struct scalarType
{
    long scalarID;           // מספר הזהות של התלמיד
    char name[20];           // שם התלמיד
    int groupNo;             // מספר קבוצת האימון שבה משובץ התלמיד - האינדקס במערך הקבוצות
    strengthPtr sp;          // מצביע לצומת בעץ הכושר של הקבוצה שבה משובץ התלמיד
                             // (עץ הכושר יתואר בהמשך)
}scolarRec, *scolarPtr;
```

להלן מבנה של צומת בעץ כושר בשפת C :

```
typedef struct strengthType
{
    int strength;             // ציון הכושר הגופני
    int IDIndex;              // מיקום התלמיד במערך התלמידים
    struct strengthType *left; // מצביע לתת-העץ השמאלי
    struct strengthType *right; // מצביע לתת-העץ הימני
}strengthRec, *strengthPtr;
```

עתה נפרט את מבנה הנתונים בעבור שדה 3 של "המבנה הראשי" – מערך דינמי של קבוצות אימון.

להלן מבנה של תא במערך זה בשפת C:

```
typedef struct groupType
{
    int  scholarNum;    // מספר התלמידים המשובצים בקבוצה
    float average;      // ממוצע הציונים של הקבוצה
    long bestScholar;   // הציון המקסימלי שקיים בקבוצה
    strengthPtr gp;     // מצביע לשורשו של עץ הכושר של הקבוצה
}groupRec, *groupPtr;
```

להלן הגדרות התקפות לכל הסעיפים שיבואו בהמשך:

```
typedef enum
{FAILURE, SUCCESS, INVALID_INPUT, ALLOCATION_ERROR
,DUPLICATE_ID} statusType;
typedef enum {FALSE,TRUE} boolean;
typedef enum {GET,PUT} modType;
```

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

<p>פונקצייה זו מקבלת שני פרמטרים: <code>num</code> – מספר שלם, ו-<code>mode</code>. אם ערכו של <code>mode</code> הוא PUT – אז הפונקצייה מחזירה את המיקום בטבלת הערבול לשם הכנסת המספר <code>num</code>. הערה: במקרה של הכנסת ערך, הנח כי תמיד יימצא מקום בטבלת הערבול. אם ערכו של <code>mode</code> הוא GET – אז הפונקצייה מחפשת בטבלת הערבול את המספר <code>num</code>. אם היא מוצאת אותו, אז היא מחזירה את מיקומו בטבלה זו; אחרת היא מחזירה את הערך <code>(-1)</code>.</p>	<p><code>int hash(long num, modType mode)</code></p>
<p>פונקצייה זו מקבלת שני פרמטרים: <code>pl</code> – כתובת של מצביע לשורשו של עץ כושר כלשהו, <code>t</code> – כתובת של מצביע לצומת (בודד), המייצג תלמיד. הפונקצייה מוסיפה לעץ הכושר את הצומת שעליו מצביע <code>t</code>, ומחזירה את הערך <code>SUCCESS</code>. הערות: 1. הנח כי התלמיד לא קיים בעץ הכושר הזה. 2. לאחר הוספת הצומת, עץ הכושר יישאר עץ AVL.</p>	<p><code>statusType insertTree(strengthPtr *pl, strengthPtr *t)</code></p>

<p>פונקצייה זו מקבלת שני פרמטרים: t – כתובת של מצביע לשורשו של עץ כושר כלשהו, p – מצביע לצומת בודד בעץ כושר זה. הפונקצייה מבצעת את הצעדים האלה:</p> <ul style="list-style-type: none"> • יוצרת צומת חדש שעליו מצביע q (מצביע עזר). צומת זה יכול את כל המידע המשוכן בצומת שעליו מצביע p. • מוציאה מעץ הכושר את הצומת שעליו מצביע p. • מחזירה מצביע לצומת שעליו מצביע q. <p>הערה: לאחר הוצאת התלמיד, עץ הכושר יישאר עץ AVL.</p>	<p>strengthPtr deleteFromStrengthTree (strengthPtr *t, strengthPtr p)</p>
<p>פונקצייה זו מקבלת את הפרמטר t, שהוא מצביע למערך דינמי של קבוצות אימון. פונקצייה זו מגדילה בתא נוסף את המערך. התא שנוסף מייצג קבוצת אימון חדשה, שהיא ריקה. נוסף על כך, הפונקצייה מגדילה ב-1 את מספר קבוצות האימון הקיימות במערכת הממוחשבת.</p>	<p>void addGroup(groupPtr *t)</p>

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

להלן הגדרות של משתנים גלובליים:

```
header head;
```

```
headPtr DS = &head;
```

```
int numOfGroups; // (נכון לרגע זה)
```


ענה על הסעיפים שלהלן:

א. לפניך פונקצייה שכותרתה:

```
statusType addScolar(long ID, char name[], int group, long strength)
```

פונקצייה זו מקבלת את הפרמטרים האלה:

ID – מספר הזהות של התלמיד,

name – שם התלמיד,

group – מספרה של קבוצת האימון שבה משובץ תלמיד זה,

strength – ציון הכושר הגופני של התלמיד.

פונקצייה זו אמורה להוסיף תלמיד חדש שמספרו ID למערכת הממוחשבת.

נוסף על כך, הפונקצייה מעדכנת את מספר התלמידים הרשומים במערכת.

הפונקצייה מחזירה ערך מטיפוס statusType כמפורט בטבלה שלהלן:

ALLOCATION_ERROR	אם קיימת בעיה בהקצאת זיכרון
DUPLICATE_ID	אם התלמיד שמספרו ID כבר קיים בטבלת הערבול
INVALID_INPUT	אם הקבוצה שמספרה group לא קיימת במערכת הממוחשבת
SUCCESS	אם התלמיד שמספרו ID נוסף בהצלחה למערכת הממוחשבת

בפונקצייה חסרים **שישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)-(6), בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
statusType addScolar(long ID, char name[], int group, long strength)
{
    statusType status = SUCCESS;

    strengthPtr p;

    int sindx;

    int scolarNum;

    float average;

    sindx = hash(ID,GET);

    if(_____ (1) _____) return DUPLICATE_ID

    sindx = _____ (2) _____ ;

    if(group < 0 || group >= numOfGroups) return INVALID_INPUT;

    DS->scolarsArr[sindx].scolarID = ID;

    strcpy(DS->scolarsArr[sindx].name,name);

    DS->scolarsArr[sindx].groupNo = group;

    p = malloc(sizeof(strengthRec));

    if(p == NULL)

        return ALLOCATION_ERROR;

    p->strength = strength;

    p->IDIndex = sindx;
```

```
_____ (3) _____ = p;  
scolarNum = DS->groupsArr[group].scolarNum;  
average = DS->groupsArr[group].average;  
average = (average*scolarNum+strength)/(scolarNum +1);  
DS->groupsArr[group].average = average;  
_____ (4) _____;  
if(strength > _____ (5) _____)  
    _____ (5) _____ = strength;  
status = insertTree(_____ (6) _____);  
DS->numOfScolars++;  
return status;  
}
```

ב. מהו זמן הריצה הממוצע של הפונקצייה המוצגת בסעיף א', כאשר k מציין את מספר התלמידים בקבוצת האימון שבה משובץ התלמיד, ו- n מציין את מספר התלמידים הרשומים במערכת הממוחשבת?

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

ג. לפניך פונקצייה שכותרתה:

```
statusType removeScolar(long scolarID)
```

פונקצייה זו מקבלת את הפרמטר `scolarID` – מספר הזהות של תלמיד.

הפונקצייה אמורה להסיר מן המערכת הממוחשבת את התלמיד שמספרו `scolarID`. במקרה זה, ערכו של `scolarID` ישתנה ל-0.

נוסף על כך, הפונקצייה מעדכנת את מספר התלמידים הרשומים במערכת זו, וכן את יתר הפרטים בעבור הקבוצה שבה היה משובץ התלמיד שמספרו `scolarID`.

שים לב: לאחר הסרת התלמיד, הקבוצה שממנה הוסר יכולה להישאר במערכת הממוחשבת כקבוצה ריקה.

הפונקצייה מחזירה ערך מטיפוס `statusType` כמפורט בטבלה שלהלן:

אם התלמיד שמספרו <code>scolarID</code> לא קיים במערכת הממוחשבת	INVALID_INPUT
אם התלמיד שמספרו <code>scolarID</code> הוסר בהצלחה מן המערכת הממוחשבת	SUCCESS

בפונקצייה חסרים **תשעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(9), בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
statusType removeScolar(long scolarID)
```

```
{  
  
    statusType status = SUCCESS;  
  
    strengthPtr t,s,q;  
  
    int indx,group;  
  
    scolarRec scr;  
  
    groupRec gpr;  
  
    long strength;  
  
    indx = hash(scolarID,GET);  
  
    if(indx == -1) return INVALID_INPUT;
```

```
DS->scolarsArr[indx].scolarID = 0;

group = _____(1)_____ ;

scr = DS->scolarsArr[indx];

gpr = DS->groupsArr[group];

strength = scr._____(2)_____ ;

if(gpr.scolarNum == 1) gpr.average = 0;

    else gpr.average = (gpr.average * gpr.scolarNum - strength)/(gpr.scolarNum -1);

    _____(3)_____ = gpr.average;

DS->groupsArr[group].scolarNum--;

s = _____(4)_____ ;

t = _____(5)_____ ;

q = deleteFromStrengthTree(&t,s);

DS->groupsArr[group].gp = t;

if(!t) DS->groupsArr[group].bestScolar=0;

else

    {

        if(q->strength == gpr.bestScolar)

            {

                while(_____(6)_____)

                    t = _____(6)_____ ;

                DS->groupsArr[group].bestScolar = _____(7)_____ ;

            }

    }

scr.scolarID = 0;

_____(8)_____ ;

_____(9)_____ ;

return status;
```

}

ד. מהו זמן הריצה הממוצע של הפונקצייה המוצגת בסעיף ג', כאשר k מציין את מספר התלמידים בקבוצת האימון שבה התלמיד משובץ, ו- n מציין את מספר התלמידים הרשומים במערכת הממוחשבת?

1. $O(k)$

2. $O(\log k)$

3. $O(1)$

4. $O(\log(n + k))$

ה. לפניך פונקצייה **רקורסיבית** שכותרתה:

```
void printTree(strengthPtr t)
```

פונקצייה זו מקבלת את הפרמטר t אשר מצביע לשורשו של עץ כושר השייך לקבוצת אימון כלשהי.

הפונקצייה מדפיסה בעבור כל תלמיד בקבוצת אימון זו את שמו ואת ציון הכושר הגופני שלו. שמות התלמידים וציוני הכושר הגופני שלהם יודפסו בצורה ממוינת, בסדר עולה, על פי ציוני כושרם הגופני.

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 printTree(strengthPtr t)
```

```
{
    scalarRec sr;
    if (t != NULL)
    {
        _____ (1) _____;
        sr = _____ (2) _____;
        printf("\n%s\t%ld\n", sr.name, sr.sp->strength);
        _____ (3) _____;
    }
}
```

}

1. מהי סיבוכיות זמן הריצה של הפונקצייה המוצגת בסעיף ה', כאשר k מציין את מספר התלמידים בקבוצה שאליה שייך עץ הכושר, ו- n מציין את מספר התלמידים הרשומים במערכת הממוחשבת?

1. $O(k)$

2. $O(\log k)$

3. $O(1)$

4. $O(\log(n + k))$

2. לפניך פונקצייה שכותרתה:

```
statusType riseScolarLevel(long ID, long strength)
```

פונקצייה זו מקבלת את הפרמטרים האלה:

ID – מספר הזהות של התלמיד,

strength – ציון הכושר הגופני.

פונקצייה זו מעדכנת עבור תלמיד שמספרו ID את ציון הכושר הגופני לציון חדש שהוא strength. נוסף על כך, הפונקצייה מעדכנת את כל המידע במערכת הממוחשבת בעבור קבוצת האימון שבה התלמיד משובץ.

הפונקצייה מחזירה ערך מטיפוס statusType כמפורט בטבלה שלהלן:

אם התלמיד שמספרו ID לא קיים במערכת הממוחשבת	INVALID_INPUT
אם התלמיד שמספרו ID הוסר בהצלחה מן המערכת הממוחשבת	SUCCESS

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 riseScolarLevel(long ID,long strength)
{
    statusType status;
    strengthPtr spr,q,t;
    int indx,group,num;
    long stch;
    long average;
    indx = hash(ID,GET);
    if(indx == -1) return INVALID_INPUT;
    spr = _____(1)_____;
    group = DS->scolarsArr[indx].groupNo;
    stch = spr->strength;
    t = DS->groupsArr[group].gp;
    q = _____(2)_____;
    q->strength = strength;
    status = _____(3)_____;
    if(_____(4)_____ < strength)
        _____(4)_____ = strength;
    average = DS->groupsArr[group].average;
    num = DS->groupsArr[group].scolarNum;
    average = (average_____(5)_____ - stch)/num;
    DS->groupsArr[group].average = average;
    return status;
}
```


ח. לפיכך פונקציית שכותרתה:

```
statusType transferScolar(long ID, int group1, int group2)
```

פונקציית זו מקבלת את הפרמטרים האלה:

ID – מספר הזהות של תלמיד,

group1 – מספר קבוצה,

group2 – מספר קבוצה.

פונקציית זו אמורה להעביר את התלמיד שמספרו ID מן הקבוצה group1 אל הקבוצה group2, ולעדכן את כל הפרטים בעבור הקבוצות group1 ו־ group2, בהתאם.

הנחות:

1. הקבוצה group1 שונה מן הקבוצה group2.
2. הקבוצה group2 קיימת במערכת הממוחשבת.

הפונקציית מחזירה ערך מטיפוס statusType כמפורט בטבלה שלהלן:

ALLOCATION_ERROR	אם קיימת בעיה בהקצאת זיכרון
INVALID_INPUT	אם התלמיד שמספרו ID אינו קיים במערכת הממוחשבת, או אם הקבוצה שמספרה group1 אינה קיימת במערכת הממוחשבת
SUCCESS	אם התלמיד שמספרו ID הועבר בהצלחה מקבוצה אחת לקבוצה אחרת

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.

```
statusType transferScolar(long ID, int group1, int group2)
{
    statusType status = SUCCESS;
    char name[20];
    int indx;
    strengthPtr s;
    long grade;
    indx = hash(ID,GET);
    if(indx == -1) return INVALID_INPUT;
    if(group1 != DS->scolarsArr[indx].groupNo)
        return INVALID_INPUT;
    strcpy(name, _____(1)_____);
    s = _____(2)_____;
    grade = s->strength;
    status = _____(3)_____;
    if(status == SUCCESS)
        status = _____(4)_____;
    return status;
}
```

ט. מהו זמן הריצה הממוצע של הפונקצייה המוצגת בסעיף ח', כאשר k_1 מציין את מספר התלמידים המשובצים בקבוצת האימון $group_1$, k_2 מציין את מספר התלמידים המשובצים בקבוצת האימון $group_2$, ו- n מציין את מספר התלמידים הרשומים במערכת הממוחשבת?

$$1. \quad O(\log k_1 \cdot k_2 \cdot n)$$

$$2. \quad O(\max(\log k_1, \log k_2))$$

$$3. \quad O(\log k_1) * O(\log k_2)$$

$$4. \quad O(\log n)$$

י. לפניך פונקצייה **רקורסיבית** שכותרתה:

```
statusType moveTree(strengthPtr *s1, strengthPtr *s2, int group)
```

כאשר:

s_1 – כתובת של מצביע לעץ כושר של קבוצה שמספרה $group$.

נסמן את העץ הזה ב- T_1 .

s_2 – כתובת של מצביע לעץ כושר של קבוצה שמספרה שונה מ- $group$.

נסמן את העץ הזה ב- T_2 .

הפונקצייה מעבירה את כל האיברים של העץ T_2 אל העץ T_1 .

נוסף על כך, בעבור כל האיברים המשובצים בעץ T_2 , הפונקצייה מעדכנת את מספר הקבוצה שלהם ל- $group$. כמו כן, היא מעדכנת את כל הפרטים בעבור הקבוצה שמספרה $group$, ובעבור הקבוצה שמספרה שונה מ- $group$.

הפונקצייה מחזירה ערך מטיפוס $statusType$ כמפורט בטבלה שלהלן:

ALLOCATION_ERROR	אם קיימת בעיה בהקצאת זיכרון
SUCCESS	אם כל האיברים של העץ T_2 הועברו בהצלחה אל העץ T_1

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.

```
statusType moveTree(strengthPtr *s1, strengthPtr *s2, int group)
{
    long ID;
    int group2;
    statusType status = SUCCESS ;
    strengthPtr t,u;
    u = *s1;
    t = *s2;
    if(t == NULL) return status;
    group2 = ( _____ (2) _____ ).groupNo;
    _____ (1) _____ = group;
    status = moveTree(&u,&(t->left),group);
    status = moveTree(&u,&(t->right),group);
    t->left = NULL;
    t->right = NULL;
    ID = ( _____ (2) _____ ).scolarID;
    status = _____ (3) _____;
    t = NULL;
    return status;
}
```

שאלה 2 – שאלת חובה (25 נקודות)

בשאלה זו ארבעה סעיפים (א'–ד'). עליך לענות על כל הסעיפים.

להלן הגדרות של קבוע, ושל צומת בעץ חיפוש בינארי.

```
#define N 500000

typedef struct nodeType
{
    int val;                // שדה מידע
    struct nodeType *left;  // מצביע לתת־העץ השמאלי
    struct nodeType *right; // מצביע לתת־העץ הימני
} nodeRec, *nodePtr;
```

כמו כן, נתונה ספרייה הכוללת הגדרות ופונקציות לטיפול במחסנית (Stack), שבה כל איבר הוא מצביע לצומת של עץ בינארי.

ההגדרות:

```
#define STACK_MAX_SIZE 500000 // גודל מקסימלי של מחסנית
typedef nodePtr stack_item;
typedef struct
{
    int top;
    stack_item data[STACK_MAX_SIZE];
} stack, *stackPtr;
```

הפונקציות:

<code>void init(stackPtr s)</code>	פונקצייה זו מאתחלת את המחסנית s להיות ריקה.
<code>void push(stackPtr s, stack_item x)</code>	פונקצייה זו מוסיפה את האיבר x למחסנית s . הנחה: המחסנית s מאותחלת ואינה מלאה.
<code>stack_item pop(stackPtr s)</code>	פונקצייה זו שולפת את האיבר שבראש המחסנית s ומחזירה את ערכו. הנחה: המחסנית s מאותחלת ואינה ריקה.
<code>stack_item top(stackPtr s)</code>	פונקצייה זו מחזירה את ערכו של האיבר שבראש המחסנית s , מבלי להוציאו מהמחסנית. הנחה: המחסנית s מאותחלת ואינה ריקה.
<code>int IsEmpty(stackPtr s)</code>	פונקצייה זו מחזירה את הערך 1 אם המחסנית s ריקה, אחרת - היא מחזירה את הערך 0. הנחה: המחסנית s מאותחלת.

כמו כן, נתונה ספריית פונקציות נוספת, המכילה, בין היתר, את הפונקציות שלהלן:

<code>void insertTree(nodePtr *pl, nodePtr *t)</code>	פונקצייה זו מקבלת את הפרמטרים האלה: pl - כתובת של מצביע לשורשו של עץ חיפוש בינארי, t - כתובת של מצביע לצומת בודד כלשהו. הפונקצייה מוסיפה את הצומת הזה לעץ החיפוש.
<code>void deleteTree(nodePtr *pl, nodePtr t)</code>	פונקצייה זו מקבלת את הפרמטרים האלה: pl - כתובת של מצביע לשורשו של עץ חיפוש בינארי, t - מצביע לצומת הקיים בעץ. הפונקצייה מוציאה מעץ החיפוש את הצומת שעליו מצביע t .

<p>nodePtr findTree(nodePtr pl, int num)</p>	<p>פונקצייה זו מקבלת את הפרמטרים האלה: pl – מצביע לשורשו של עץ חיפוש בינארי, num – מספר שלם. הפונקצייה מחפשת את הערך num בעץ החיפוש. אם הערך num קיים בעץ, אזי הפונקצייה מחזירה מצביע לצומת הזה; אחרת – הפונקצייה מחזירה את הערך NULL.</p>
<p>void merge(int a[], int b[], int c[], int m, int n)</p>	<p>פונקצייה זו מקבלת שני מערכים: a ו-b, שגודלם m ו-n בהתאמה, וכל אחד מן המערכים ממוין בסדר עולה. הפונקצייה ממזגת את המערכים a ו-b למערך c, כך שהמערך c יהיה ממוין בסדר עולה.</p>
<p>void my_sort(int arr[], int N)</p>	<p>פונקצייה זו מקבלת את המערך arr ואת גודלו N, וכל תא במערך זה מכיל מספר שלם. הפונקצייה ממיינת את איברי המערך בסדר עולה.</p>

הנח שהפונקציות האלו כתובות וניתן להשתמש בהן בכל הסעיפים הבאים, בלי לכתוב אותן מחדש. כמו כן, בעבור כל סעיף, ניתן להשתמש בכל פונקצייה שמומשה בסעיפים שלפניו. להלן הגדרות של משתנים גלובליים התקפות לכל הסעיפים שיבואו בהמשך:

```
int a[N];  
int b[N];  
int c[2*N];  
int i=0, j=0, k;  
nodePtr t1=NULL, t2=NULL, t3=NULL;  
nodePtr p;  
int h;
```

ענה על הסעיפים שלהלן:

א. לפניך פונקצייה **רקורסיבית** שכותרתה:

```
void scanTree(nodePtr p, int arr[])
```

פונקצייה זו מקבלת את שני הפרמטרים האלה:

p – מצביע לשורשו של עץ חיפוש בינארי,

arr – מערך ריק.

הפונקצייה מעתיקה למערך arr את איברי העץ שלשורשו מצביע p , כך שאיברי המערך arr יהיו ממוינים בסדר עולה.

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 scanTree(nodePtr p, int arr[])
{
    if(p != NULL)
    {
        static int i = 0;
        _____ (1) _____;
        _____ (2) _____;
        _____ (3) _____;
    }
}
```

ב. לפניך פונקצייה **לא רקורסיבית** שמבצעת בדיוק את אותה הפעולה שמבצעת הפונקצייה הרקורסיבית שבסעיף א' (היא מעתיקה למערך `arr` את איברי העץ שלשורשו מצביע `p`, כך שאיברי המערך `arr` יהיו ממוינים בסדר עולה).
כותרת הפונקצייה היא:

```
void scanTree2(nodePtr p, int arr[])
```

פונקצייה זו מקבלת את שני הפרמטרים האלה:

`p` – מצביע לשורשו של עץ חיפוש בינארי,

`arr` – מערך ריק.

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.

```
void scanTree2(nodePtr p, int arr[])
{
    nodePtr t;
    stackPtr S;

    s = malloc(sizeof(int)*STACK_MAX_SIZE);
    init(S);
    t = p;
    do
    {
        while (t != NULL)
        {
            push(S, t);
            _____(1)_____;
        }
    }
}
```

```
if (!IsEmpty(S))  
{  
    t = _____ (2) _____;  
    arr[j] = _____ (3) _____;  
    j++;  
    t = _____ (4) _____;  
}  
} while (_____ (5) _____);  
}
```

ג. לפניך הגדרות:

עץ בינארי למהדרין הוא עץ בינארי, אשר הדרגה של כל צומת בו היא 0 או 2, כלומר כל צומת בו הוא או עלה או שיש לו בדיוק שני בנים.

עץ בינארי כמעט שלם (almost complete binary tree) הוא עץ בינארי למהדרין, שעבורו קיים מספר שלם אי-שלילי, k , כך שיתקיימו שני התנאים האלה:

1. כל העלים בעץ הם ברמה ה- k או ברמה ה- $k+1$.

2. אם לצומת בעץ יש צאצא ימני שהוא עלה ברמה ה- $k+1$ אז כל העלים מבין צאצאיו השמאליים, שהם גם עלים, יהיו גם הם ברמה ה- $k+1$.

לפניך פונקצייה **רקורסיבית** שכותרתה:

```
nodePtr createTree(nodePtr t, int h)
```

פונקצייה זו מקבלת את שני הפרמטרים האלה:

t – מצביע לעץ ריק,

h – מספר שלם.

הפונקצייה בונה עץ בינארי כמעט שלם שגובהו h , ובעל N צמתים, כאשר N הוא משתנה גלובלי. שדה המידע של כל צומת בעץ לא מאוחסן. הנח כי גובה העץ הוא $O(\log N)$.

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 createTree(nodePtr t, int h)
```

```
{
    static int i = 1;
    if (h >= 0 && i <= N)
    {
        t = malloc(sizeof(nodeRec));
        _____(1)_____;
        t->left = NULL;
        t->right = NULL;
        t->left = _____(2)_____;
        t->right = _____(3)_____;
    }
    return t;
}
```

ד. נתונה פונקציית **רקורסיבית** שכותרתה:

```
void TR2 (nodePtr t, int d[])
```

הפונקציית מקבלת את הפרמטרים האלה:

t – מצביע לשורשו של עץ **בינארי כמעט שלם** בעל $2 \cdot N$ צמתים, ושדה המידע בכל צומת לא מאותחל. נסמן את העץ הזה ב- $T1$.

d – מערך ממיון בסדר עולה, בגודל $2 \cdot N$.

הפונקציית משכנת את איברי המערך d בעץ $T1$ באופן כזה שהעץ $T1$ יהיה עץ **חיפוש** בינארי.

נוסף על כך, נתונים שני עצי AVL בשם $t1$ ו- $t2$, וכל אחד מהם בעל N צמתים, כאשר ערכו של N נתון.

לפניך אלגוריתם (המוצג בעמוד 30), בעל סיבוכיות זמן ריצה $O(N)$, אשר ממזג את שני עצי ה- $t1$ AVL ו- $t2$ לעץ AVL אחד – $t3$, בעל $2 \cdot N$ צמתים.

האלגוריתם משתמש בפונקציית הרקורסיבית $TR2$ המתוארת לעיל.

בפונקציית $TR2$ ובאלגוריתם חסרים יחד **תשעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)–(9), בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

להלן גוף הפונקציית $TR2$:

```
void TR2 (nodePtr t, int d[])
{
    static int i = 0;
    if (t == NULL || _____ (1) _____) return;
    else
    {
        _____ (2) _____;
        _____ (3) _____;
        _____ (4) _____;
    }
}
```

להלן האלגוריתם למיזוג העצים:

$t3 = \text{NULL};$

_____ (5) _____;

_____ (6) _____;

_____ (7) _____;

$h = \lceil \log_2(N * 2 + 1) - 1 \rceil;$

$t3 =$ _____ (8) _____;

_____ (9) _____;

פרק שני (35 נקודות)

ענה על אחת מבין השאלות 3–4 (לכל שאלה – 35 נקודות).

שאלה 3

בשאלה זו 15 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 ב'.

בסעיפים א'–ד' התייחס לבעיה שלהלן:

יש להציע מבנה נתונים עבור הקבוצה S , כאשר S מתקבלת מאיחודן של m קבוצות זרות של מספרים ממשיים, כלומר $S = \bigcup_{i=1}^m S_i$, וכל S_i (לכל i , $1 \leq i \leq m$) היא קבוצה בת n איברים לכל היותר.

מאחר ש- S מתקבלת מאיחודן של m קבוצות זרות, מובן שכל האיברים שב- S שונים זה מזה.

מבנה הנתונים הנדרש יתמוך בפעולות שלהלן:

פעולה	תיאור הפעולה
Insert(i,k)	הוספת האיבר k לקבוצה S_i שקיימת במבנה הנתונים
Delete(i,k)	מחיקת האיבר k מן הקבוצה S_i שקיימת במבנה הנתונים. ניתן להניח כי k שייך לקבוצה S_i
Min_Max()	מציאת האיבר המינימלי מבין: $M_1, M_2, \dots, M_i, \dots, M_m$, כאשר M_i הוא האיבר בעל הערך המקסימלי שבקבוצה S_i (לכל i , $1 \leq i \leq m$)
GlobalMax()	מציאת האיבר המקסימלי מבין כל האיברים שבקבוצה S

הערה: המבנה יכול להיות מורכב מכמה מבנים פשוטים יותר.

סטודנט הציע את מבנה הנתונים שלהלן כפתרון לביצוע **יעיל** של כל הפעולות לעיל.

להלן הצעת הסטודנט:

מבנה הנתונים, שמתחזק קבוצה S של מספרים ממשיים שונים, יורכב מהמבנים שלהלן:

1. עץ AVL1, שהוא עץ חיפוש בינארי מאוזן (עץ AVL), הממוין לפי האינדקסים של הקבוצות. כל צומת x בעץ AVL1 מייצג את אחת הקבוצות S_i , וצומת זה מכיל:
 - שדה key אשר מכיל את אינדקס הקבוצה (אחד מבין הערכים: $1, 2, \dots, m$)
 - שדה $left$ – מצביע ל**בן שמאלי**
 - שדה $right$ – מצביע ל**בן ימני**
 - שדה $point$ – מצביע לשורשו של עץ AVL **אחר**, שהוא עץ חיפוש בינארי מאוזן (עץ AVL), שמשוכנים בו האיברים השייכים לאותה הקבוצה. כלומר, אם ערכו של השדה key הוא i , אז השדה $point$ יצביע על שורשו של עץ AVL אחר, שמשוכנים בו כל האיברים השייכים לקבוצה S_i בלבד.
2. עץ AVL2, שהוא עץ חיפוש בינארי מאוזן (עץ AVL). עץ זה מכיל את האיבר המקסימלי מכל קבוצה. כלומר, עץ AVL2 יכיל את: $M_1, M_2, \dots, M_i, \dots, M_m$, כאשר M_i הוא האיבר בעל הערך המקסימלי בקבוצה S_i (לכל $i, 1 \leq i \leq m$).
3. משתנה $MMax$ – מכיל את האיבר המקסימלי הקיים בעץ AVL2.
4. משתנה $GMax$ – מכיל את האיבר המינימלי הקיים בעץ AVL2.

א. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה $Insert(i,k)$ או הפעולה $Delete(i,k)$?

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

ב. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה $\text{Min_Max}()$?

1. $O(\log n)$

2. $O(\log m)$

3. $O(1)$

4. $O(\log n + \log m)$

ג. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה $\text{GlobalMax}()$?

1. $O(\log n)$

2. $O(1)$

3. $O(\log m)$

4. $O(\log n + \log m)$

ד. שני תלמידים התבקשו להציע הצעות למימוש הפעולה $\text{Insert}(i,k)$.
הצעתו של תלמיד א':

צעד 1: חפש צומת שבו נמצא הערך i בעץ AVL1 . צומת זה יסומן ב- p .

צעד 2: הכנס את k לעץ AVL שלשורשו מצביע השדה point של הצומת p , וסיים.

הצעתו של תלמיד ב':

צעד 1: חפש צומת שבו נמצא הערך i בעץ AVL1 . צומת זה יסומן ב- p .

צעד 2: הכנס את k לעץ AVL שלשורשו מצביע שדה point של הצומת.

צעד 3: אם k הוא המקסימום בקבוצה S_i , אז מחק מהעץ AVL2 את M_i , כאשר M_i הוא מפתח בעל הערך המקסימלי בקבוצה S_i , וסיים.

איזו מבין שתי ההצעות מממשת במלואה את הפעולה $\text{Insert}(i,k)$?

1. אף אחת מבין שתי ההצעות

2. הצעתו של תלמיד א' בלבד

3. הצעתו של תלמיד ב' בלבד

4. שתי ההצעות גם יחד

סעיפים ה'–י' מתייחסים לבעיה שלהלן:

חברת הבנייה "מבנים בע"מ" רוצה לשמור את נתוני משכורות העובדים ולבצע עליהם כמה פעולות. נניח שנתוני משכורות העובדים נמצאים בקבוצה S , והחברה מעסיקה n עובדים, כלומר, $|S| = n$. יש להציע מבנה נתונים בגודל $O(n)$ התומך בפעולות האלה:

פעולה	תיאור הפעולה
Init(employees, median_key3)	פעולה זו מאתחלת את המבנה. employees היא רשימה לא ממוינת של n העובדים והמשכורות שלהם. median_key3 הוא השלישון של משכורות העובדים, והוא משתנה בהתאם לעדכונים (להגדרת השלישון, ראה הערה בתחתית הטבלה).
Insert(k)	פעולה זו מוסיפה עובד שמשכורתו k למבנה הנתונים. הפעולה שומרת על התכונות של מבנה הנתונים.
Remove_Min()	פעולה זו מוציאה את העובד בעל המשכורת המינימלית. אם יש יותר מאחד כזה, הפעולה תוציא אחד מהם. הפעולה שומרת על התכונות של מבנה הנתונים.
Remove_Max()	פעולה זו מוציאה את העובד בעל המשכורת המקסימלית. אם יש יותר מאחד כזה, הפעולה תוציא אחד מהם. הפעולה שומרת על התכונות של מבנה הנתונים.
Average()	פעולה זו מדפיסה את המשכורת הממוצעת של עובדי החברה.
Median3()	פעולה זו מדפיסה את השלישון של משכורת העובדים בחברה.

הערה:

שלישון – הערך המופיע במקום ה- $\lceil \frac{n}{3} \rceil$ לאחר מיון האיברים בסדר עולה, כלומר האיבר ה- $\lceil \frac{n}{3} \rceil$ בגודלו (ערך המיקום ה- $\lceil \frac{n}{3} \rceil$).

סטודנט הציע את מבנה הנתונים הבא כפתרון לביצוע **יעיל** של כל הפעולות לעיל.

המבנה יכול את המבנים שלהלן:

• ארבע ערימות:

- Heap1 – ערימת מקסימום אשר תכיל את הנתונים של כל העובדים
- Heap2 – ערימת מינימום אשר תכיל את הנתונים של כל העובדים
- Heap3 – ערימת מקסימום אשר תכיל נתונים של $\left\lceil \frac{n}{3} \right\rceil$ העובדים. ערימה זו תכיל את $\left\lceil \frac{n}{3} \right\rceil$ האיברים הקטנים ביותר
- Heap4 – ערימת מינימום אשר תכיל נתונים של $\left(n - \left\lceil \frac{n}{3} \right\rceil \right)$ העובדים

הערות: כל איבר שנמצא ב-Heap3 קטן מכל איבר שנמצא ב-Heap4.

בכל אחת מן הערימות העדיפות נקבעת על-פי שכר העובד.

- משתנה size אשר יכול את מספר העובדים
- משתנה Salary_sum אשר יכול את סכום המשכורות של כלל העובדים
- מבנה (רשומה) של עובד אשר יכול שדות בסיסיים וגם נתונים, כגון:

- מיקומו בערימת המקסימום Heap1
- מיקומו בערימת המינימום Heap2
- באיזו ערימה הוא נמצא מבין הערימות Heap3 או Heap4
- מיקומו בערימה Heap3 או Heap4

שים לב: בפעולות Insert או Remove (הכנסת איבר או הוצאתו), ניתן להעביר איבר מערימה אחת לערימה אחרת במידת הצורך (מ-Heap3 ל-Heap4 וכן להיפך), כדי לשמור על דרישותיו של הסטודנט.

על סמך מבנה הנתונים שהציע הסטודנט, ענה על השאלות שבסעיפים ה'–ט"ו.

ה. בחר את ההיגד הנכון מבין ההיגדים הבאים:

1. השלישון יהיה תמיד האיבר המקסימלי מבין האיברים שקטנים משורשה של הערימה
Heap1
2. השלישון יהיה תמיד האיבר המקסימלי מבין האיברים שגדולים משורשה של הערימה
Heap2
3. השלישון יהיה תמיד האיבר המינימלי של הערימה Heap4
4. השלישון יהיה תמיד האיבר המקסימלי של הערימה Heap3

ו. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה
Median3() ?

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

ז. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה
Insert(k) ?

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

ח. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה

? `Remove_Min()`

1. $O(\log \log n)$

2. $O(\log n)$

3. $O(1)$

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

ט. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה

? `Init(employees, median_key3)`

1. $O(n)$

2. $O(\log n)$

3. $O(1)$

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

י. על סמך מבנה הנתונים שהציע הסטודנט, מהי סיבוכיות זמן הריצה למימוש הפעולה

? `Average()`

1. $O(1)$

2. $O(\log n)$

3. $O(n)$

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

בסעיפים י"א-ט"ו התייחס למימוש הפעולה `Init(employeees, median_key3)` שלהלן:

צעד 0: קליטת הנתונים ל-S וקביעת ערכם של `size` ושל `Salary_sum`.

צעד 1: העתק את אברי הקבוצה S למערך העזר A1.

צעד 2: העתק את אברי הקבוצה S למערך העזר A2.

צעד 3: _____(1)_____

צעד 4: _____(2)_____

צעד 5: `temp =` _____(3)_____

צעד 6: `partition(S, temp, _____(4)_____, _____(5)_____)`

צעד 7: בנה ערימה Heap3 מהאיברים המשוכנים במערך A3.

צעד 8: בנה ערימה Heap4 מהאיברים המשוכנים במערך A4.

הערה: `partition(S, z, S1, S2)` היא פונקצייה המעתיקה את הסדרה S לשתי סדרות, S1 ו-S2, כך שאיבריה יחולקו באופן הזה: לסדרה S1 יועתקו האיברים הקטנים מ-z או השווים לו, ולסדרה S2 יועתקו האיברים הגדולים מ-z. הסדרות S1 ו-S2 אינן בהכרח ממוינות.

במימוש הזה 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 ב'.

י"א. הביטוי החסר (1) הוא:

1. בנה ערימה Heap1, מאברי המערך A1, ובעבור כל עובד שמור את מיקומו בערימה זו.

2. העתק את אברי הקבוצה S למערך העזר A3.

3. בצע פעולת `partition` ביחס לערך הראשון שבמערך העזר A1.

4. בצע פעולת `partition` ביחס לערך שמיקומו $\left\lfloor \frac{n}{3} \right\rfloor$ במערך העזר A1.

י"ב. הביטוי החסר (2) הוא:

1. בנה ערימה Heap2, מאברי המערך A2, ובעבור כל עובד שמור את מיקומו בערימה זו.
2. העתק את אברי הקבוצה S למערך העזר A4.
3. בצע פעולת partition ביחס לערך הראשון שבמערך העזר A2.
4. בצע פעולת partition ביחס לערך שמיקומו $\left\lfloor \frac{n}{3} \right\rfloor$ במערך העזר A2.

י"ג. הביטוי החסר (3) הוא:

1. $\text{select}(n, S)$
2. $\text{select}\left(\left\lfloor \frac{n}{3} \right\rfloor, S\right)$
3. $\text{select}\left(\left\lfloor \frac{n}{2} \right\rfloor, S\right)$
4. $\text{select}(1, S)$

י"ד. הביטוי החסר (4) הוא:

1. A1
2. A2
3. A3
4. S

ט"ו. הביטוי החסר (5) הוא:

1. A1
2. A2
3. A4
4. S

שאלה 4

בשאלה זו 15 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 ב'.

סעיפים א'–ה' מתייחסים לבעיה שלהלן:

יהי $G = (V, E)$ גרף לא מכוון, שהוא עץ.

המרחק בין שני קודקודים בגרף G הוא אורך המסלול הקצר ביותר הקיים ביניהם (בספירת קשתות).

הקוטר של גרף G הוא המרחק המקסימלי מבין כל המרחקים (המסלולים המינימליים) שבין כל זוגות הקודקודים האפשריים בגרף G .

לפניך אלגוריתם **יעיל** ככל האפשר שמוצא את הקוטר של גרף G .

האלגוריתם

צעד 1: בחר קודקוד v בעץ באופן שרירותי.

צעד 2: הרץ את האלגוריתם _____ (1) _____ החל מן הקודקוד v .

צעד 3: יהי קודקוד u שעבורו _____ (2) _____.

צעד 4: הרץ את האלגוריתם _____ (3) _____ החל מן הקודקוד u .

צעד 5: יהי קודקוד t שעבורו _____ (4) _____.

צעד 6: המרחק בין שני הקודקודים t ו- u בגרף הוא **הקוטר** של הגרף.

באלגוריתם הזה חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. בכל אחד מן הסעיפים שלהלן, בחר את הביטוי החסר מבין ארבע האפשרויות הנתונות, והקף בעיגול את הספירה המייצגת אותה בדף התשובות שבנספח ב'.

א. הביטוי החסר (1) הוא:

1. BFS
2. מיון טופולוגי
3. DFS
4. דייקסטרה למציאת המסלול הקצר ביותר

ב. הביטוי החסר (2) הוא:

1. התקבל המרחק המינימלי מן הקודקוד v
2. התקבל המרחק המקסימלי מן הקודקוד v
3. התקבל מעגל שאורכו מקסימלי
4. התקבל מעגל שאורכו זוגי

ג. הביטוי החסר (3) הוא:

1. BFS
2. מיון טופולוגי
3. DFS
4. דייקסטרה למציאת המסלול הקצר ביותר

ד. הביטוי החסר (4) הוא:

1. התקבל המרחק המינימלי מן הקודקוד u
2. התקבל מעגל שאורכו זוגי
3. התקבל מעגל שאורכו מקסימלי
4. התקבל המרחק המקסימלי מן הקודקוד u

ה. סיבוכיות זמן הריצה של האלגוריתם הנתון היא:

1. ריבועית, כפונקצייה של גודל הקלט
2. מעריכית, כפונקצייה של גודל הקלט
3. $O(|V|)$ בהכרח
4. $O(|V| \log |V|)$

ו. $T(n) = 27T(n/3) + 9n^3$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(n^4 \log n)$
2. $\Theta(n^6 \log n)$
3. $\Theta(n^3 \log n)$
4. $\Theta(n^3)$

ז. $T(n) = 3T(n/9) + \sqrt{n} \log^2 n$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

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

ח. $T(n) = 4T(n-1) - 3T(n-2) + 8n^2 + 5n - 7$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(n^2)$

2. $\Theta(3^n)$

3. $\Theta(n^2 3^n)$

4. $\Theta(n 3^n)$

ט. לפניך קטע קוד:

```
s = 0;
for(t = 1; t <= n; t = t*2)
{
    for(j = 1; j <= t; j++)
    {
        s++;
    }
}
```

מהי סיבוכיות זמן הריצה של קטע הקוד הנתון כפונקציה של n ?

1. $\Theta(n)$

2. $\Theta(n \log n)$

3. $\Theta(n^2)$

4. $\Theta(n \log \log n)$

י. לפניך קטע קוד:

```
for(t = n; t >= 1; t = t/3)
{
    a = 1;
    while(a < t)
        a = a*3 ;
    while(a > 3)
        a = a/3;
}
```

מהי סיבוכיות זמן הריצה של קטע הקוד הנתון כפונקציה של n ?

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

2. $O(\log \log \log n)$

3. $O(\log n)$

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

י"א. $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + 1$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $T(n) = \Theta(n)$

2. $T(n) = \Theta(n^{\log_5 4})$

3. $T(n) = \Theta(n \log n)$

4. $T(n) = \Theta\left(n^{\frac{4}{5}}\right)$

י"ב.

נתונות שתי פונקציות מונוטוניות עולות f ו- g ($g: \mathbb{N} \rightarrow \mathbb{N}$, $f: \mathbb{N} \rightarrow \mathbb{N}$).

לטענת הסטודנט A: "אם $f(n) = \Theta(g(n))$, אז בהכרח $\log(f(n)) = \Theta(\log(g(n)))$ ".

לטענת הסטודנט B: "אם $f(n) = \Theta(g(n))$, אז בהכרח $2^{f(n)} = \Theta(2^{g(n)})$ ".

בחר בהיגד הנכון מבין ההיגדים שלהלן:

1. שני הסטודנטים צודקים בטענותם
2. רק הסטודנט A צודק בטענתו
3. רק הסטודנט B צודק בטענתו
4. אף אחד מן הסטודנטים אינו צודק בטענתו

י"ג.

לפניך קטע קוד:

```
void func(int n)
{
    int i, j = 0;
    if(n < 1) return;
    for(i = 1; i < n; i *= 2) j++;
    func(n - 1);
}
```

מהי סיבוכיות זמן הריצה של קטע הקוד הנתון כפונקצייה של n ?

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

י"ד. $T(n) = 2T(n/2) + n \log_2 \left(\frac{n^2}{4} \right)$ היא פונקציית זמן הריצה של אלגוריתם מסוים, הפועל על קלט שגודלו n . מהי סיבוכיות זמן הריצה של האלגוריתם?

1. $\Theta(n \log n)$

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

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

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

ט"ו. **טענה א':** לשתי הפונקציות $\{2^{\log^2 n}, n^{\log n}\}$ יש התנהגות אסימפטוטית זהה.

טענה ב': אם פונקציית זמן הריצה של אלגוריתם הפועל על קלט שגודלו n היא: $T(n) = 16T(\sqrt[4]{n}) + \log^2 n$, אזי $O((\log n) \cdot \log \log n)$.

בחר בהיגד הנכון מבין ההיגדים שלהלן:

1. שתי הטענות נכונות

2. אף טענה אינה נכונה

3. רק טענה ב' נכונה

4. רק טענה א' נכונה

בהצלחה!

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

הקף בעיגול את הספרה המייצגת את התשובה הנכונה לכל סעיף.

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

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

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