מבוא למדעי המחשב מ'/ח' (234114/7)

תרגיל בית 5

על נושא המייל להתחיל במספר הקורס (234114/234117) והתרגיל ולהמשיך בנושא השאלה שתופיע בגוף המייל, לדוגמא

234114 hw5 Question no. 1 ..

- **טרם הפנייה, בדקו בדף ה-FAQ של התרגיל אם השאלה שלכם כבר נענתה.**

הנחיות כלליות:

- הגשה ב**בודדים**. עליכם לכתוב את הפתרונות לבד ולהגיש ביחידים.
- קראו את השאלות בעיון לפני שתתחילו בפתרונן.
- הקפידו לתעד את הקוד שלכם בהערות באנגלית.
- שאלות ותשובות נפוצות בנוגע לתרגיל יתפרסמו באתר כל כמה זמן תחת סעיף F.A.Q- **חובה להיכנס ולהתעדכן! כל דגש שמפורסם שם הוא מחייב!**
- מלבד מילואים, לא יתקבלו תרגילים אחרי מועד הגשה. הגשה באיחור לאחר מועד הגשה נחשבת כאי-הגשה.
- כל יום מילואים = יום דחייה. על מנת לקבל את הדחייה, עליכם לשלוח באי-מייל, עותק של האישור המראה שהייתם במילואים (טופס 3010). אם האישור יגיע אליכם בתאריך מאוחר, יש להודיע על כך למתרגל האחראי לפני תאריך הגשת התרגיל.
- ערעורים ניתן להגיש עד שבוע לאחר קבלת הציון.
- **לא ניתן לערער על תוצאות הבדיקה האוטומטית.**
- **שימו לב! הבדיקה הינה בחלקה אוטומטית, ולכן הקפידו להדפיס בדיוק בפורמט שהתבקשתם ובידקו עם DiffMerge את הפלט שלכם מול הפלט של הדוגמאות שקיבלתם.**
- השתמשו באתר הבדיקה העצמית.
- ההגשה הינה אלקטרונית ו**בבודדים** דרך אתר הקורס. קובץ ההגשה יהיה מסוג **zip** (ולא אף פורמט אחר) ויכיל בתוכו את הקבצים הבאים בלבד, ללא כל תיקיות:
  - קובץ **students.txt** עם מספר תעודת הזהות שלך וכתובת האי-מייל שלך.
  - קבצי פתרון **hw5q1.c hw5q2.c, hw5q3.c**
  - **חובה לשמור את אישור ההגשה (ולא רק את קוד האישור!!) שמקבלים מהמערכת לאחר שמגישים, עד לסיום הקורס.**
  - יש להקפיד להגיש את כל הקבצים בדיוק עם השמות שמופיעים לעיל. הגשה שלא תעמוד בתנאי זה **לא תתקבל ע"י המערכת!** אם המערכת לא מקבלת את התרגיל שלכם, חפשו את הפתרון לבעיה באתר הקורס תחת הכפתור FAQ.

שאלה מס' 1: מיונים (המלצה – מומלץ לפתור בעצמיכם, אבל אין להגיש)

יש לכתוב תכנית אשר קולטת מספר שלם ואי שלילי k ומערך a של מספרים שלמים שמקיימים את החוקיות הבאה: המערך a אינו בהכרח ממוין, אבל כל מספר ב – a נמצא במרחק של לכל היותר k תאים ממיקומו במערך לאחר מיון. לדוגמא:

- אם k=0, המערך a בהכרח ממוין (כל מספר נמצא במרחק 0 ממיקומו במערך הממוין, כלומר נמצא בדיוק במקום)
- אם k=1 מערך a יכול למשל להיות בעל הערכים הבאים:

2	1	3	4	7	20	15	22	30	28

- אם k=2 מערך a יכול למשל להיות בעל הערכים הבאים:

1	-2	-4	2	10	5	12	7	100	101

על התכנית למיין את מערך a באופן יעיל ככל הניתן (יש להשתמש בתכונה המיוחדת של המערך). ניתן להניח שהמספר k קבוע ביחס לגודל המערך. לבסוף התכנית תדפיס את המערך הממוין.

פירוט הקלט והפלט:

1. תודפס הודעה המבקשת להזין מספר k אי שלילי:

Please enter a non negative integer (k):

2. ייקלט הערך של k. אם הקליטה הצליחה והערך של k תקין (אי שלילי), התכנית תמשיך לשלב הבא. אם הקליטה הצליחה אך k שלילי, תודפס ההודעה של שלב 1 שוב, וייקלט ערך חדש עבור k, עד שיתקבל ערך תקין או שאחת הקליטות תיכשל. אם קליטה נכשלת, התכנית תסתיים מיידית (אין צורך להדפיס הודעה על כך).

3. תודפס הודעה המבקשת להזין את גודל המערך n, מספר אי שלילי:

Please enter a non negative integer (n):

4. ייקלט הערך של n, הטיפול בערכים שליליים ובקליטה שנכשלה יהיה זהה לתיאור של שלב 2 (כאשר במקרה של קליטה חוזרת תודפס ההודעה המתאימה משלב 3).

5. תודפס הודעה המבקשת להזין את תוכן המערך:

Please enter n integer values with the special characteristic:

6. ייקלטו n מספרים שלמים אשר מקיימים את התכונה המתוארת לעיל. המספרים בקלט יהיו מופרדים ע"י רווח\ים או שורות חדשות. אם קליטה נכשלת, התכנית תסתיים מיידית (אין צורך להדפיס הודעה על כך).
ניתן להניח שאם הקליטה הצליחה, רצף המספרים מקיים את התכונה, אין צורך לבדוק זאת.

7. הערכים יודפסו באופן ממויין.

שאלה 2: **Backtracking** (חובה)

עליכם לכתוב תכנית אשר בהינתן מספר מנדטים שקיבלה כל מפל גה, ואילו מפלגות מוכנות לשבת ביחד בקואליציה, סופרת כמה קואליציות ניתן ליצור.

נסמן ב – N את מספר המפלגות (מוגדר ב – define). כתבו פונקציה שחתימתה:

```
int count_coalitions(int mandates[N], bool can_collaborate[N][N])
```

אשר מחשבת ומחזירה את מספר הקואליציות האפשריות בהינתן:

- מערך mandates בגודל N, כך ש – mandates[i] מכיל את מספר המנדטים שקיבלה המפלגה ה – i.
- מערך can_collaborate - מטריצה שמכילה רק אפסים ואחדות ואומרת לכל שתי מפלגות האם הן מסכימות להיות חברות באותה קואליציה (can_collaborate[i][j]=1 אם ורק אם המפלגות i ו-j מסכימות לשבת יחדיו בקואליציה). ניתן להניח כי מטריצה זו תהיה תמיד סימטרית (can_collaborate[i][j]=can_collaborate[j][i]) ועל האלכסון הראשי יהיה תמיד 1 (can_collaborate[i][i]=1).

קואליציה תיקרא "אפשרית" אם כל המפלגות המרכיבות אותה מסכימות לשבת אחת עם השנייה (לפי המטריצה can_collaborate) ובנוסף, סכום המנדטים של כל המפלגות החברות בה גדול ממש מ-60. אם אין אף קואליציה אפשרית, הפונקציה תחזיר 0.

דוגמא: אם ישנן N=5 מפלגות (ממוספרות מ-0 עד 4), מערך mandates נראה כך (אינדקסים באדום):

20	***14***	***30***	***16***	***40***
0	***1***	***2***	***3***	***4***

והמטריצה can_collaborate נראית כך (אינדקסים בצבע אדום, תוכן בצבע שחור):

	0	***1***	***2***	***3***	***4***
0	***1***	***1***	***1***	***0***	***1***
1	***1***	***1***	***1***	***1***	***1***
2	***1***	***1***	***1***	***1***	***0***
3	***0***	***1***	***1***	***1***	***1***
4	***1***	***1***	***0***	***1***	***1***

הקואליציות האפשריות הן:
{1, 3, 4} (גודל 70)
{0, 1, 4} (גודל 74)
{0, 1, 2} (גודל 64)
הקבוצה {0,4} אינה מהווה קואליציה אפשרית כי גודלה בדיוק 60 (ולא גדול ממש מ-60).

הקבוצה {1,2,4} אינה מהווה קואליציה אפשרית כי מפלגות 2 ו-4 לא מסכימות לשבת אחת עם השנייה (can_collaborate[2][4]==0).

לכן, הפונקציה תחזיר 3. **אין צורך לכתוב תכנית ראשית.** עליכם להוריד מאתר הקורס את הקובץ hw5q2_main.c המכיל פונקציית main הקולטת קלט בפורמט הנדרש, קוראת לפונקציה count_coalitions עם הפרמטרים כפי שנקלטו ומדפיסה את מס' הקואליציות האפשריות לפי ערך ההחזרה. כל שעליכם לעשות הוא להוסיף לקובץ זה את המימוש שלכם של הפונקציה count_coalitions, לשמור מחדש את הקובץ בשם hw5q2.c ולצרף אותו להגשה.

דגשים נוספים:

- על הפתרון ליישם backtracking כפי שנלמד בכיתה
- אין דרישות סיבוכיות, אולם על הפונקציה לשאוף "לגזום ענפים" ככל שניתן (כלומר לא לבצע קריאות רקורסיביות מיותרות)
- ניתן להשתמש בחתימה הנתונה כפונקציית מעטפת ולממש פונקציית עזר עם פרמטרים נוספים. כמו כן, ניתן להיעזר בפונקציות עזר נוספות כרצונכם.

שאלה 3: **Backtracking** (המלצה – מומלץ לפתור בעצמיכם, אבל אין להגיש)

בשאלה זו יש לממש וריאציה נוספת של שאלה 2. כעת, במקום לספור את מספר הקואליציות האפשריות, עליכם לחשב את מספר המפלגות המשתתפות בקואליציה המקסימלית (סכום מקסימלי של כמות המנדטים של המפלגות המשתתפות) ולהחזיר את האינדקסים של המפלגות המשתתפות בה ממוינים בסדר עולה (בלי להשתמש באלגוריתם מיון, צרו את הפלט ממוין).

נסמן ב – N את מספר המפלגות (מוגדר ב – define). כתבו פונקציה שחתימתה:

```
int max_coalition(int mandates[N], bool can_collaborate[N][N],
                  int max_parties[N])
```

אשר מחזירה (ערך ההחזרה של הפונקציה) את מספר המפלגות המשתתפות בקואליציה המקסימלית, ואת האינדקסים של המפלגות המשתתפות בה במערך max_parties. אם למשל הקואליציה המקסימלית מורכבת מ – k מפלגות, ערך ההחזרה יהיה k, ו – k התאים הראשונים במערך max_parties יכילו את אינדקסים המפלגות המשתתפות בקואליציה המקסימלית, ושאר התאים במערך מכילים ערכים כלשהם (חסרי משמעות).

המערכים mandates ו – can_collaborate, וכמו כן התנאים להיווצרות קואליציה, מוגדרים בשאלה 2.

עבור המערכים לדוגמא משאלה 2, ערך ההחזרה יהיה 3, ובסיום ריצת הפונקציה max_parties יכיל את הערכים הבאים:

0	***1***	***4***	***??***	***??***
0	***1***	***2***	***3***	***4***

אין צורך לכתוב תכנית ראשית. עליכם להוריד מאתר הקורס את הקובץ hw5q3_main.c המכיל פונקציית main הקולטת קלט בפורמט הנדרש, קוראת לפונקציה max_coalition עם הפרמטרים כפי שנקלטו ומדפיסה את הקואליציה המקסימלית. כל שעליכם לעשות הוא להוסיף לקובץ זה את המימוש שלכם של הפונקציה max_coalition, לשמור מחדש את הקובץ בשם hw5q3.c ולצרף אותו להגשה.

דגשים נוספים:

- על הפתרון ליישם backtracking כפי שנלמד בכיתה
- אין דרישות סיבוכיות, אולם על הפונקציה לשאוף "לגזום ענפים" ככל שניתן (כלומר לא לבצע קריאות רקורסיביות מיותרות)
- ניתן להשתמש בחתימה הנתונה כפונקציית מעטפת ולממש פונקציית עזר עם פרמטרים נוספים. כמו כן, ניתן להיעזר בפונקציות עזר נוספות כרצונכם.

התכוננות למבחן, לפני שתתחילו לפתור מבחנים

- ודאו שאתם יודעים לתכנת כל מיון שלמדנו בעל פה ובעצמיכם. הכונה בכך היא שתדעו לתכנת אותם מהראש לגמרי (כמו במצב של מבחן) בלי להסתכל בשקפים. הכונה היא לכל המיונים, bubblesort, maxsort, quicksort, mergesort, bucketsort. ודאו שאתם מבינים את הסיבוכיויות השונות של כל אחד, ואיך כל אחד עובד (כמה השוואות, כמה גישות למערך, עומק הרקורסיה, סיבוכיות של צומת בעץ הרקורסיה, איך עץ הרקורסיה נראה...), הכל ברמת השקפים, לא פחות ולא חובה יותר.
- ודאו שאתם יודעים לתכנת חיפוש בינארי בעל פה (כמו בכדור הקודם), ולפחות גוון אחד של החיפוש, כלאמר, שאלה אחת מהשקפים מהתרגולים או מההרצאות על חיפוש בינארי.
- לדעת היטב מאוד את נושא המחרוזות והמצביעים. לקרוא את שקפי התרגולים פעם אחת לפחות. לפתור שאלה שמערבת מיון ביחד עם מחרוזות, או מיון של מערך דו ממדי, או מיון מסוים של מצביעים, מאוד חשוב.

כשתגיעו לפתור מבחנים, לפתור הרבה שאלות סיבוכיות (אלה שאלות מתנה במבחן, לא לטעות במקומות האלה אחר כך כי חבל), ואחר כך לפתור הרבה מאוד בקטרקינג, כי הרבה מהקושי והלחץ במבחן נופלים על השאלה הזאת, ושאלות נוספות לפי הצרכים שלכם. אם למשל אתם מרגישים שאתם חלשים בנושא מסוים, למשל שאלות שמערבות מחרוזות ומצביעים, פתרו שאלות בו. אני מציע שחילוק השאלות יהיה חצי מהזמן בקטרקינג, עד שתבינו את הקטע בשאלות כאלה, וחצי מהזמן שאלות אחרות. אחר כך עשו באוות נפשכם.

אני אדגיש, שהכוונה בפתרון שאלה היא תכנות פתרון מלא ועובד על מחשב.

אני ממליץ כמו כן לודא שאתם תמיד מטפלים בבעיות שקשורות לנכונות קלט והקצית זכרון. כלאמר, אם קולטים באמצעות scanf, לודא שמספר המשתנים שנקלטו בהצלחה הוא כבמחרוזות, ואם לא, לחזור על הקליטה של scanf לפי ניתוח הערך המוחזר, או פשוט יציאה exit(ERROR) מהתכנית (כש-ERROR הוגדר ב-DEFINE), או בדיקת נכונות הקצית זכרון:

```
double **ptr = (double **) malloc(sizeof(double*) * n);

if(ptr == NULL)

exit(ERROR);

for(int i =0 ; i < n; i++)

{

ptr[i] = (double *) malloc(sizeof(double) * m);

if(ptr[i] == NULL)

exit(ERROR);

}

//פה קורים דברים

for(int i =0 ; i < n; i++)

{

free(ptr[i]);

}
```

```
free(ptr);
```

זכרו שמאלוק לא מאתחלת לנו את הזכרון המוקצה לאפס, כך שאם לכך תדרשו בתכנית, תצטרכו לעשות גם זאת. קטע הקוד לעיל הוא רק דוגמה. תוכלו במצב הזה להשתמש בפונקציה calloc (חפשו עליה בגוגל), או לעשות לולאה שמאתחל את הזכרון. חשוב לזכור לשחרר זכרון תמיד כשהוא לא נמצא בשימוש, וחשוב לדאוג שלא תהינה חריגות מהזכרון המוקצה לתכנית שלנו, כמו למשל, אם אנחנו מדפיסים מערך צ'ארים באיזו-שהיא צורה, שאין בסופו '\0'.

בנוגע לאתחול מערכים ומערכים דו ממדיים, לא תהיינה בעיות פה כפי שהיו בבדיקת תרגילי הבית, כלאמר, לא תחויבו לאתחל אותם.

בהצלחה!