שאלה 4 (25 נקודות)

בשאלה זו נרצה לפתור את בעיית **מגדלי הוואי**. בדומה לבעיית מגדלי הנוי, בבעיה יש שלושה מגדלים (נסמנם 1, 2 ו-3), ומספר טבעות המונחות על אחד המגדלים. ואולם כיאה להוואי, ובניגוד לבעיית מגדלי הנוי, הטבעות הן צבעוניות: מכל גודל טבעת יש שתי טבעות, האחת אדומה והשנייה לבנה, ובסה"כ יש m טבעות אדומות ו-m טבעות לבנות.

במצב ההתחלתי הטבעות ממוקמות זו מעל זו במגדל מס' 1 ומסודרות לפי גודלן (הגדולה ביותר למטה). מכל זוג טבעות באותו הגודל, זו הלבנה ממוקמת מתחת לזו האדומה. למשל, עבור m=3 המצב ההתחלתי הוא:

כללי הבעיה נותרים כמו בבעיה המקורית:

- אין להניח טבעת גדולה מעל טבעת קטנה ממנה. עם זאת, **מותר** להניח שתי טבעות מאותו הגודל האחת מעל לשנייה, ללא קשר לצבע (כלומר מותר לבנה על אדומה וגם אדומה על לבנה מאותו הגודל).
- אין להזיז יותר מטבעת אחת בו זמנית.

בשאלה זו אין הגבלה על סיבוכיות הפונקציות. לשם הזזת הטבעות, ניתן להשתמש בפונקציות הבאות:

```
void movered(int from, int to) {
  printf("move red ring from %d to %d\n", from, to);
}
```

```
void movewhite(int from, int to) {
  printf("move white ring from %d to %d\n", from, to);
}
```

סעיף א

כתבו פונקציה שמקבלת את m – מספר זוגות הטבעות במגדל המקור (כלומר סה"כ ישנן 2m טבעות, שתיים בכל גודל), וכן את מספר מגדל המקור ומספר מגדל היעד. הפונקציה מדפיסה את ההוראות להעברת כל הטבעות ממגדל המקור למגדל היעד, תוך שימוש במגדל השלישי כמגדל עזר. במצב הסופי, על הטבעות להיות **באותו הסדר** כמו שהיו במגדל המקור – כלומר לבנה מתחת לאדומה. לדוגמה, עבור m=1, מגדל המקור 1 ומגדל היעד 3, הפונקציה תדפיס:

```
move red ring from 1 to 2
move white ring from 1 to 3
move red ring from 2 to 3
```

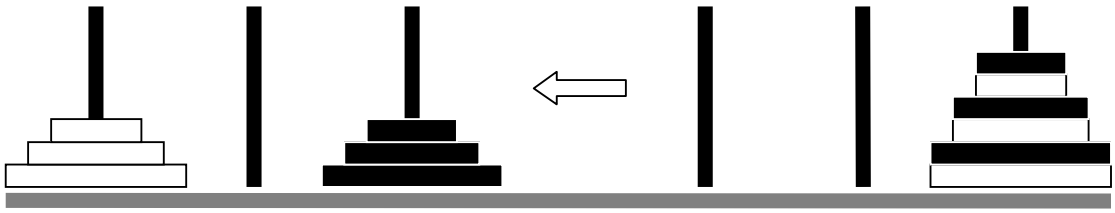
```
void hawaii1(int m, int from, int to) {
   int via=1+2+3-from-to;
   if (m<=0) return;
   hawaii1(m-1,from,to);
   move_red(from,via);
   move_white(from,via);
   hawaii1(m-1,to,from);
   move_white(via,to);
   move_red(via,to);
   hawaii1(m-1,from,to);
}
```

הסבר:
יש מספר פתרונות אפשריים, ואנו נציג כאן שניים. הפתרון הראשון פועל על צמדי טבעות במשותף ולא מפריד אותן במהלך פעולתו, והפתרון השני מאפשר פיצול כזה. הפתרון הראשון יעיל יותר בכמות ההעברות בשל שימוש ב-3 קריאות ל-hawaii1 בגודל m-1 בעוד שהשני מחייב 4 קריאות כאלה. למרות זאת, שני פתרונות אלו קבילים ונחשבים למענה מלא לשאלה.

```
void hawaii1(int m, int from, int to) {
   int via=1+2+3-from-to;
   if (m<=0) return;
   hawaii1(m-1,from,to);
   move_red(from,via);
   hawaii1(m-1,to,via);
   move_white(from,to);
   hawaii1(m-1,via,from);
   move_red(via,to);
   hawaii1(m-1,from,to);
}
```

סעיף ב

כתבו פונקציה שמקבלת את m – מספר זוגות הטבעות במגדל המקור, וכן את מספר מגדל המקור ומספר מגדל היעד. הפונקציה מדפיסה הוראות להעברת הטבעות, כך שבסוף התהליך הטבעות מופרדות על פי צבען – כל הלבנות נמצאות במגדל היעד (הגדולה למטה) וכל האדומות נמצאות במגדל המקור (הגדולה למטה). המצב ההתחלתי והמצב הסופי נראים כך:



לדוגמה, עבור m=1, מגדל המקור 1 ומגדל היעד 3, הפונקציה תדפיס:

```
move red ring from 1 to 2
move white ring from 1 to 3
move red ring from 2 to 1
```

בסעיף זה ניתן להשתמש בפונקציה מהסעיף הקודם, גם אם לא מימשתם אותה.

```
void hawaii2(int m, int from, int to) {
   int via=1+2+3-from-to;
   if (m<=0) return;
   hawaii1(m-1,from,to);
   move_red(from,via);
   hawaii1(m-1,to,via);
   move_white(from,to);
   hawaii1(m-1,via,to);
   move_red(via,from);
   hawaii1(m-1,to,from);
   hawaii2(m-1,from,to);
}
```

הסבר:
גם כאן קיימים מספר פתרונות אפשריים, ואנו נתמקד באחד מהם שנראה הכי טבעי. בפתרון זה אנו מעבירים m-1 צמדים לעמודת to, מעבירים את האדום ל-via ומעליו שמים את כל הטבעות הקטנות שהיו ב-to. בשלב זה הטבעת הלבנה הגדולה ביותר נמצאת ב-from ויכולה לעבור ל- to- יעדה הסופי. לאחר מכן, m-1 צמדי הטבעות הקטנות מועברות ל- to על מנת לאפשר לטבעת האדומה הגדולה לעבור ליעדה הסופי (from). בסיום יש להחזיר את כל הטבעות הקטנות למקומן המקורי ב-from. אסור לשכוח לקרוא בשלב זה ל-hawaii2 על מנת להמשיך תהליך רקורסיבי שיפעל על m-1 הצמדים הקטנים.

פתרון אחר מתבסס על העברת כל הטבעות (ע"י hawaii1) ל-to, כך שהטבעת הלבנה הגדולה נמצאת במקומה. לאחר מכן יש להעביר m-1 צמדים ל-via, כך שהטבעת האדומה הגדולה תוכל להתמקם ב-from – מיקומה הסופי. שוב נדרש להעביר את m-1 הצמדים הקטנים ל-from, והתהליך ימשיך באופן רקורסיבי עליהם.

משפחת פתרונות שלישית שאותה נזכיר בקצרה מציעה מבנה בו hawaii2 אינו מבוסס רקורסיה, אלא על לולאת for או while. בפתרון זה מאתחלים את התהליך בהעברת כל הטבעות ל-via. לאחר מכן, פועלים בלולאה מ-i=0 ועד m-1, וכנגד כל ערך של i נעביר m-i צמדים ל-to (אז הטבעת הלבנה שבתחתית נמצאת במקומה הרצוי), m-i+1 מאלה ל-via, ואז את האדומה שב-to נעביר ל- from – מקומה הסופי. בשלב זה בעמודת via ממתינים m-i+1 צמדי טבעות לטיפול דומה, ולכן הלולאה.