שאלה 3 (25 נקודות)

בעיית מספרי הנוי דומה מאוד לבעיית מגדלי הנוי: ישנם 3 מוטות ו-n טבעות המסודרות בהתחלה על אחד המוטות, מהגדולה (בתחתית) לקטנה (בפסגה). ההבדלים:

- **הטבעות ממוספרות**: מספר הטבעת הגדולה ביותר n ומספר הטבעת הקטנה ביותר 1
- **המגדלים ממוספרים**: לכל אחד משלושת המגדלים יש מספר (0, 1 או 2)
- **המצב הסופי** הדרוש הוא שכל טבעת תהיה ממוקמת על המוט שמספרו הוא כשארית חלוקת מספר הטבעת ב-3. כלומר הטבעות 3,6,9... על מוט מספר 0, טבעות 1,4,7... על מוט מספר 1 וטבעות 2,5,8... על מוט מספר 2.

חוקי המשחק נשמרים:

1. מותר להזיז רק טבעת בודדת בכל צעד
2. אסור למקם טבעת מעל לטבעת קטנה ממנה

לנוחותכם, מובא להלן הקוד של בעיית הנוי המקורית, כפי שנלמד בכיתה:

```
typedef enum{A,B,C} tower_t;

void hanoi(int n, tower_t from, tower_t to)
{
  tower_t via = (A + B + C) - from - to;
  if (n == 0) return;
  hanoi(n-1, from, via);
  printf("Move disc %d from %c to %c\n",
                  n, 'A' + from, 'A' + to);
  hanoi(n-1, via, to);
}
```

סעיף א'

כתבו פונקציה **רקורסיבית** הפותרת את בעיית מספרי האנוי עם **הקלה**: חוק מספר 1 לעיל מתבטל, כלומר **ניתן** להזיז מספר טבעות עליונות בבת אחת (תוך שמירה על הסדר היחסי שלהן) ממגדל למגדל. הזזה של מספר טבעות ניתן לעשות אך ורק ע"י קריאה לפונקציה `move_several` אשר חתימתה:

```
void move_several(int n, tower_t from, tower_t to)
```

על הפונקציה שלכם לגרום להדפסת סדרת ההזזות הדרושה. שימו לב שהפונצקיה `move_several` מדפיסה הודעה המתאימה להזזות שהיא מבצעת, ולכן רק במקרה בו אתם מזיזים טבעת שלא ע"י קריאה ל-`move_several` עליכם להדפיס הודעה מתאימה.

```
void hanoi2(int n, int loc)
{
```

סעיף ב'

כתבו פונקציה רקורסיבית הפותרת את בעיית מספרי הנוי תוך קיום כל הדרישות, בפרט דרישה 1 לעיל.
רמז: חשבו במה ניתן להחליף את הקריאה לפונקציה `move_several`.

```
void hanoi3(int n, int loc)
{
```