מטלה 4 :

שאלה 2:

מגיש: ליאל ברניקר

|הסבר כללי על התשובה בשאלה:

סעיף א':
בסעיף א' מימשתי את הפונקציה אשר מקבלת מערך דו ממיידי המייצג את החפצים והערכים שיש לכל שחקן בהתאמה לחפץ.

לדוגמא:
Arr = [[1, 1], [11, 11], [13, 13]]

כאן ניתן לראות שישנם שני שחקנים ושלושה חפצים , כל שורה מייצגת חפץ והערכים שכל שחקן משייך לו. כל עמודה מייצגת שחקן והערך שנותן למספר החפץ לפי השורה.
בדוגמא לחפץ הראשון שחקן אחד נותן שווי של 1 וגם שחקן 2 נותן לחפץ הראשון שווי של 1.

באופן כללי התבקשתי בסעיף זה בשאלה להתייחס למקרה בו ישנם רק שני שחקנים ושהערכות שלהם זהות. אך רשמתי פונקציות כלליות אשר ניתן להכניס מספר לא קבוע של שחקנים ומספר שאינו קבוע של חפצים לחלק לשחקנים אלו.
חיפוש במרחב המצבים לבעיית חלוקת חפצים בין שני שחקנים עם הערכות זהות מוכל בפתרון הפונקציה שכתבתי לכן הפונקציה שרשמתי אכן מקיימת את החיפוש בצורה הנכונה.

תחילה יצרתי class אשר מייצג מצב , מכיל את הערכים הנוכחיים לכל שחקן וכמה משאבים חולקו עד אותו מצב זה .
בסעיף א הפונקציה מקבלת מערך דו ממדי כמו שתיארתי, לאחר מכן יוצרת מצב התחלתי, אשר במצב זה כל ערכי השחקנים מאותחלים ל0. לאחר מכן הפונקציה קוראת לפונקציה פנימית רקורסיבית אשר רצה כל עוד יש חפצים לחלק ויוצרת את כל המצבים האפשריים. שומרת את כל המצביים הסופיים ומוצאת מי הוא המצב בעל הערך המינימאלי המקסימאלי.
האלגוריתמים עובד לפי השלבים בתיאור במצגת, ללא שלב הגיזום

state-space search

מצב של חלוקה חלקית := וקטור באורך n+1 (מספר החפצים שחולקו, הערך של כל שחקן).

המצב של חלוקה ריקה = (0, ..., 0; 0).

הרעיון:

- נתחיל מחלוקה ריקה;
- ניצור את כל n המצבים הנובעים ממצב קיים + חלוקת חפץ אחד;
- נמחק מצבים מיותרים (גיזום – pruning; פירוט בהמשך);
- מתוך כל המצבים הסופיים (= m חפצים חולקו), נבחר מצב עם הערך המינימלי הגדול ביותר.

הפונקציה מחזירה רשימה המכילה את כל המצבים הסופיים ואת המצב עם הערך המינימאלי הגדול

```
return [finalStates, maxOfMin(finalStates)]
```
ביותר.
הדרך שבה בחרתי ליצור את המצבים הינה dfs

בסעיף ב' ביצעתי את אותו האלגוריתם אך הוספתי את שלב הגיזום. בחרתי את גיזום כלל א'

גיזום (pruning) – כלל א

כלל א: נמחק מצבים זהים.

ניתן לראות כי בבדיקות שרשמתי בmain (בדיקות assert) אכן נראה כי בהוספת שלב הגיזום מספר המצבים הסופיים קטן בהרבה לאומת הרצת האלגוריתם ללא כלל הגזירה נראה כמה השוואות:
מצב 1:

```
Arr = [[2, 2], [2, 2]]
assert len(findAllStates(Arr, 2, 2)[0]) is 4
```

```
assert len(findAllStatesWithChop(Arr, 2, 2)[0]) is 3
```

בהרצה זו ניתן לראות שנקבל פחות מצבים סופיים באלגוריתם המשתמש בכלל הגיזום
3<4

מצב 2:

```
Arr = [[1, 1], [6, 6], [3, 3], [4, 4], [10, 10],[1,1],[8,8],[5,5]]
assert len(findAllStates(Arr, 2, 8)[0]) is 256
```

```
assert len(findAllStatesWithChop(Arr, 2, 8)[0]) is 39
```

בהרצה זו ניתן לראות שנקבל פחות מצבים סופיים באלגוריתם המשתמש בכלל הגיזום
39<256

בנוסף ביצעתי בדיקות לראות שאכן התוכנית תחזיר את המצב עם הערך המינימאלי הגדול ביותר

* פירוט נוסף ניתן לראות בהערות המפרטות על הפונקציות השונות ובבדיקות השונות