מס טופס:_____מספר ת"ז:_____

חורף, מועד ב, 5.3.2023

בחינה בקורס תכנות בפייתון למהנדסים 0509-1820

יוני ברג, רוני קון, מיכל קליינבורט, איתן ברון, דניאל גליקמן, חגי לוי,
גיל לוונטל, אביה עשהאל, אורן קציר

הנחיות כלליות

- משך הבחינה שלוש שעות.
- חומר עזר מותר: דף A4 כתוב בשני הצדדים.
- במבחן שלוש שאלות הכוללות סעיפי משנה. במבחן 105 נקודות, כל ציון מעל 100 יעוגל ל-100. ניקוד כל סעיף מצוין לידו. אין בהכרח קשר בין ניקוד הסעיף ובין רמת הקושי שלו.
- את הפתרונות יש לכתוב במסגרות המסומנות לכל שאלה בטופס הבחינה.
- במחברת הטיוטא יש להשתמש כטיוטא בלבד. בבדיקת המבחן, מחברות הטיוטא לא ייבדקו.
- מומלץ לכתוב פתרון מלא במחברת הטיוטא ורק לאחר מכן להעתיק את הפתרון לטופס הבחינה.
- בכל סעיף ניתן להשתמש בקוד שהתבקשתם לכתוב בסעיפים קודמים, גם אם לא כתבתם אותו.
- ניתן להוסיף פונקציות עזר אם לא צוין אחרת בשאלה.
- ניתן להניח שהקלט לקוד שתכתבו תקין, אלא אם נכתב אחרת בשאלה.
- על כל סעיף ניתן לענות "איני יודע/ת את התשובה"; על סעיף זה יינתנו 20% מהנקודות. במקרה זה אין להוסיף הסבר.
- אין להשתמש בחבילות או מודולים, אלא אם נאמר במפורש. כאשר אתם מתבקשים להשתמש בחבילה אין צורך לבצע import.
- יש לרשום את מספר תעודת הזהות ואת מספר הטופס בכל עמוד למקרה שחלק מהדפים יתלשו.
- מומלץ לקרוא כל שאלה עד סופה, על כל סעיפיה, לפני תחילת הפתרון.

בהצלחה!

לשימוש הבודקים:

תעודת זהות	
מספר טופס	
שאלה 1	
שאלה 2	
שאלה 3	
ציון	

שאלה 1 (אלגוריתמים ורקורסיה) – 35 נקודות.

סעיף א (8 נקודות)

ממשו את הפונקציה is_special_prime(num) שמקבלת מספר שלם חיובי, num, ומחזירה את הערך הבוליאני True אם שני התנאים הבאים מתקיימים:

1. num הוא מספר ראשוני (כלומר מתחלק רק ב-1 ובעצמו).
2. קיים $m < 50$ טבעי כלשהו כך ש- $num = 2^m - 1$.

אחרת, הפונקציה תחזיר את הערך הבוליאני False.

דוגמאות:

```
>>> is_special_prime(7) # 7 is a prime and 2^3 - 1 = 7.
True
>>> is_special_prime(11) # 11 is a prime but 11 is not of the form 2^m - 1.
False
>>> is_special_prime(15) #15 is not a prime
False
>>> is_special_prime(31) #31 is a prime and 2^5 - 1 = 31
True
```

```
def is_special_prime(num):
```

סעיף ב (8 נקודות)

ממשו את הפונקציה common_letter_before_punct(txt) המקבלת מחרוזת (txt) לא ריקה. עליכם להניח כי המחרוזת מכילה רק אותיות קטנות, רווחים ואת סימני הפיסוק: , (פסיק) .(נקודה) ! (סימן קריאה). בנוסף ניתן להניח כי לפני סימן פיסוק, תמיד תיהיה אות (ולא רווח).

על הפונקציה ל**החזיר** את האות שמופיעה הכי הרבה פעמים לפני סימן פיסוק. ניתן להניח שהמחרוזת לא ריקה, שהיא מכילה אותיות וסימני פיסוק, ושיש אות יחידה שהופיעה מספר מקסימלי של פעמים לפני סימן פיסוק.

דוגמאות:

```
>>> common_letter_before_punct('we are learning. programming, in python. is the best')
```

```
'g'
```

הסבר: האות g מופיעה פעמיים לפני סימן פיסוק (פעם אחת לפני נקודה ופעם שנייה לפני פסיק). חוץ ממנה, רק n מופיעה לפני סימן פיסוק, אך היא מופיעה פעם אחת בלבד.

```
>>> common_letter_before_punct('students. and! teachers.')
```

```
's'
```

```
def common_letter_before_punct(txt):
```

סעיף ג (7 נקודות)

ממשו את הפונקציה sum_max_two(lst) באופן **רקורסיבי**. הפונקציה מקבלת רשימה של מספרים טבעיים. על הפונקציה להחזיר את סכום שני המספרים הגדולים ביותר ברשימה. הניחו כי הרשימה מכילה לפחות שני מספרים ושכל מספר מופיע פעם אחת בלבד.

שימו לב שיש לממש **פונקציה זו באופן רקורסיבי** ופתרון שאינו רקורסיבי לא ייקבל ניקוד. בנוסף אסור להשתמש בפונקציות max, min, ו- sort ואסור לממש פונקציות עזר אשר מחשבות sort\max\min.

```
>>> sum_max_two([12, 90])
102
>>> sum_max_two([1, 9, 34, 12, 8, 20])
54
```

```
def sum_max_two(lst):
```

סעיף ד (12 נקודות)

ממשו את הפונקציה **הרקורסיבית** exists_good_subset(lst, P, W) המקבלת רשימה (lst), ושני מספרים שלמים חיובים (P ו-W).

הרשימה lst מכילה tuple-ים כאשר כל אחד מהם מכיל שני איברים (אתם יכולים להניח שהרשימה אינה ריקה). על הפונקציה להחזיר True אם קיימת קבוצה של זוגות ברשימה כך שהסכום של כל האיברים הראשונים בזוגות בקבוצה גדול או שווה ל- P וסכום האיברים השניים בזוגות קטן או שווה ל- W. אחרת, הפונקציה תחזיר False.

עליכם לממש את הפונקציה ולהוסיף ממואיזציה.

```
>>> exists_good_subset([(50,4), (50,4)], 90, 10)
True
>>> exists_good_subset([(50,4), (50,4)], 90, 7)
```

False

>>> exists_good_subset([(60,4), (50,4), (60, 2)], 90, 7))

True

הסבר: בדוגמא הראשונה סכום שני האיברים הראשונים הוא 100 שזה גדול מ-90 וסכום שני האיברים השניים הוא 8 שקטן מ-10. בדוגמא השנייה, מצד אחד סכום האיברים השניים משני הזוגות גדול מ-7 ומצד שני, אם ניקח רק זוג אחד, סכום של האיברים הראשונים יהיה 50 שקטן מ-90. בדוגמא השלישית, הזוג הראשון והשלישי עונה על השאלה.

```
def exists_good_subset(lst, P, W, mem=None):
```

שאלה 2 - תכנות מונחה עצמים (35 נקודות)

בשאלה זו אין צורך לבצע בדיקות תקינות על הקלט.

ניקוד מלא יינתן לפתרונות שיעשו שימוש נכון בתכנות מונחה עצמים לצורך צמצום שכפול קוד

סעיף א (6 נקודות)

ממשו את המחלקה Location המייצגת מיקום. המחלקה תכלול את המתודות הבאות:

1. __init__(self, lat, lon)

מתודת הבנאי (constructor) תאתחל אובייקט חדש הכולל את השדות הבאים:

- lat: מספר עשרוני (float) אי-שלילי המייצג את קואורדינטת קו הרוחב (latitude) של המיקום הספציפי
- lon: מספר עשרוני (float) אי-שלילי המייצג את קואורדינטת קו האורך (longitude) של המיקום הספציפי

2. __repr__(self)

מתודה זו תחזיר מחרוזת המייצגת את האובייקט בפורמט (lat,lon) על-פי הדוגמא המוצגת בהמשך.

3. __eq__(self, other)

מתודה זו תחזיר True אם self הוא מיקום זהה למיקום של other ואחרת היא תחזיר False. כאשר נבדוק שוויון בין מיקומים לא נרצה לדרוש שוויון מוחלט, אלא נסתפק בבדיקה "בקירוב" של הקורדינטות – נבדוק האם הערך השלם של ה-lat והערך השלם של ה- lon של self שווים לערך השלם של ה-lat והערך השלם של ה- lon של האובייקט other. ראו דוגמא בהמשך.

- הערך השלם של מספר עשרוני הוא המספר השלם המתקבל ע"י עיגול למטה.
- זכרו שהאופרטור == מפעיל את המתודה __eq__.
- בסעיף זה מותר (אך לא חייב) להשתמש בספרייה math.

<u>דוגמה לשימוש במחלקה:</u>

```
>>> l1 = Location(32.08, 34.78)
>>>  l1
(32.08,34.78)

>>> l2 = Location(32.0, 34.0)
>>> l2 == l1 # 32.08 is “close” to 32.0, and 34.78 is “close” to 34.0
True

>>> l3 = Location(48.85, 2.35)
>>> l1 == l3
False

>>> l4 = Location(32.0, 35.0) # l1.lon=34.78 and is not “close” to 35
>>> l1 == l4
False
```

```
class Location():

    def __init__(self, lat, lon):




    def __eq__(self, other):









    def __repr__(self):
```

סעיף ב (4 נקודות)

ממשו את המחלקה Suitcase המייצגת מזוודה. המחלקה תכלול את המתודות הבאות:

1. __init__(self, owner, weight, destination)
   מתודת הבנאי (constructor) תאתחל אובייקט חדש הכולל את השדות הבאים:
   a. owner: מחרוזת (string) המתארת את שם הבעלים של המזוודה.
   b. weight: מספר שלם (int) המתאר את משקל המזוודה. המשקל תמיד נתון בקילוגרמים.
   c. destination: אובייקט מסוג Location המתאר את היעד אליו אמורה להגיע המזוודה.

2. __repr__(self)

מתודה זו תחזיר מחרוזת המייצגת את האובייקט לפי דוגמת הפלט המוצגת בהמשך. המחרוזת תכיל את שם הבעלים של המזוודה, את משקלה ואת מיקום היעד של המזוודה.

<u>דוגמה לשימוש במחלקה:</u>

```
>>> paris = Location(48.86, 2.35)
>>> s1 = Suitcase("Danny Li", 26, paris)
>>> s1
owner: Danny Li
weight: 26KG
destination: (48.86,2.35)
```

```
class Suitcase:

    def __init__(self, owner, weight, destination):









    def __repr__(self):
```

סעיף ג (10 נקודות)

ממשו את המחלקה TrackableSuitcase המייצגת מזוודה הניתנת לאיתור. TrackableSuitcase היא סוג של Suitcase ולכן תכיל את כל השדות שמכילה Suitcase ובנוסף שני שדות נוספים:

- gps_loc – אובייקט מסוג Location המייצג את מיקום המזוודה. בעת איתחול האובייקט שדה זה מייצג את מיקום המוצא של המזוודה (כלומר, המיקום שבו התחיל המעקב אחרי המזוודה).
- is_lost - משתנה בוליאני (bool) שמאותחל להיות False בעת יצירת האובייקט, ובהמשך יכול לשנות את ערכו בהתאם להאם המזוודה הלכה לאיבוד.

על המימוש להיעשות תוך הנחה שקיים בידכם הקוד של סעיף ב'.

על המחלקה לכלול את המתודות הבאות:

1. __init__(self, owner, weight, destination, gps_loc)
מתודת הבנאי (constructor) תאתחל אובייקט חדש.
2. __repr__(self)
מתודה זו תחזיר מחרוזת המייצגת את האובייקט בדומה לסעיף ב' כאשר בסופה מופיעים גם ערכי השדה החדש gps_loc. בנוסף אם המזוודה הלכה לאיבוד היא תוסיף למחרוזת את המילה "!!!LOST" כפי שניתן לראות בדוגמת הפלט בהמשך.
3. upon_arrival(self, cur_location)
מתודה זו מעדכנת את השדות הרלוונטים בעת הנחיתה בשדה התעופה:
   a. המתודה תעדכן את השדה gps_loc להיות cur_location (המיקום הנוכחי)
   b. המתודה תעדכן את is_lost להיות True אם המזוודה הלכה לאיבוד, כלומר אם המיקום הנוכחי אינו שווה למיקום היעד של המזוודה (על פי הקריטריון לשוויון מיקומים שהוגדר בסעיף א׳). אחרת, המתודה תעדכן את is_lost להיות False.
   c. הפונקציה אינה מחזירה ערך.
4. calc_distance_from_destination(self)
מתודה זו תחזיר 0 אם המזוודה לא הלכה לאיבוד ואת המרחק מה destination אם המזוודה הלכה לאיבוד. נניח לשם הפשטות שמרחק בין 2 מיקומים (lat_1, lon_1), (lat_2, lon_2) מחושב ע"י הנוסחה :
$\sqrt{(lat_1 - lat_2)^2 + (lon_1 - lon_2)^2}$
שוב, תוכלו להשתמש בספרייה math אך ניתן לפתור גמם לא שימוש בה.

<u>דוגמה לשימוש במחלקה:</u>

אתחול מזוודה שיוצאת מ cur_loc אל עבר היעד dest:

```
>>> dest = Location(32.08, 34.78)
>>> cur_loc = Location(12.23, 14.16)
>>> s4 = TrackableSuitcase("Danny Li", 34, dest , cur_loc)
>>> s4
owner: Danny Li
weight: 34KG
destination: (32.08,34.78)
current location: (12.23,14.16)
```

המזוודה נוחתת בטעות ביעד הלא נכון wrong_dest :

```
>>> wrong_dest = Location(48.86, 2.35)
>>> s4.upon_arrival(wrong_dest)
>>> s4
```

```
owner: Danny Li
weight: 34KG
destination: (32.08,34.78)
current location: (48.86,2.35)
LOST!!!

>>> s4.is_lost
True

>>> s4.calc_distance_from_destination ()
36.51401511748605
```

למרבה המזל המזוודה ממריאה שוב ונוחתת ביעד הנכון dest :

```
>>> s4.upon_arrival(dest)
>>> s4
owner: Danny Li
weight: 34KG
destination: (32.08,34.78)
current location: (32.08,34.78)

>>> s4.is_lost
False
```

```
class TrackableSuitcase(Suitcase):

    def __init__(self, owner, weight, destination, gps_loc):









    def __repr__(self):









    def upon_arrival(self, cur_location):

```

```
    def calc_distance_from_destination(self):
```

סעיף ד (7 נקודות)

ממשו את הפונקציה sort_by_weight(baggages) המקבלת רשימת מזוודות ומחזירה רשימה ממוינת של **שמות** הבעלים של המזוודות, כאשר השמות ממוינים לפי משקלי המזוודות שלהם, מהקלה ביותר לכבדה ביותר. אתם יכולים להניח שכל משקלי המזוודות שונים.

<u>דוגמה:</u>

```
>>> paris = Location(48.86, 2.35)
>>> s1 = Suitcase("Danny Li", 26, paris)
>>> s2 = Suitcase("Yann LaBlack", 18, paris)
>>> s3 = Suitcase("David Bau", 30, paris)
>>>  baggages = [s1, s2, s3]
>>>  sort_by_weight(baggages)
['Yann LaBlack', 'Danny Li', 'David Bau']
```

```
def sort by weight(baggages):
```

סעיף ה (8 נקודות)
ממשו את הפונקציה calc_avg_distance_lost_baggages(baggages) המקבלת רשימה של מזוודות. הרשימה יכולה להכיל גם מזוודות מסוג Suitcase וגם מזוודות מסוג TrackableSuitcase. הפונקציה תחזיר את ממוצע המרחקים של המזוודות שהלכו לאיבוד מהיעד שלהן.

דוגמה:
המזוודות של דני, בן, לינדה וקייט יוצאות מתל אביב לפריז:

```
>>> paris = Location(48.86, 2.35)
>>> tel_aviv = Location(32.08, 34.78)
>>> s1 = Suitcase("Danny Li", 26, paris)
>>> s2 =TrackableSuitcase("Ben Gold", 34, paris , tel_aviv)
>>> s3 = TrackableSuitcase("Linda Miller", 20, paris , tel_aviv)
>>> s4 = TrackableSuitcase("Kate Li", 20, paris , tel_aviv)
```

המזוודות של בן ולינדה הן מזוודות שניתנות לאיתור ולכן ניתן לדעת שהן נחתו ביעדים הלא נכונים:

```
>>> wrong_dest1 = Location(40.42, 3.7)
>>> s2.upon_arrival(wrong_dest1)
>>> s2.calc_distance_from_destination()
8.547286118997068
```

```
>>> wrong_dest2 = Location(45.42, 4.7)
>>> s3.upon_arrival(wrong_dest2)
>>> s3.calc_distance_from_destination()
4.1660652899348545
```

המזוודות של קייט גם היא ניתנת לאיתור ונחתה ביעד הנכון, בפריז:

```
>>> s4.upon_arrival(paris)
```

```
>>> baggages = [s1, s2, s3, s4]
```
לפיכך המזוודות של בן ולינדה הן המזוודות שישתתפו בחישוב הממוצע:

```
>>> calc_avg_distance_lost_baggages(baggages)
6.356675704465961
```

המזוודה של דני היא לא מזוודה הניתנת לאיתור ולכן היא לא משתתפת בחישוב הממוצע.

```
def calc_avg_distance_lost_baggages(baggages):
```

שאלה 3 (IO, NUMPY & Image Processing, Pandas) – 35 נקודות

סעיף א' (7 נקודות)

ממשו את הפונקציה merge_files(infile1, infile2, lst1, lst2, outfile) המקבלת נתיבים לשני קבצים המכילים רק אותיות ורווחים (infile1, infile2), שתי רשימות של מספרים טבעיים (lst1, lst2) ונתיב לקובץ שיהיה קובץ הפלט (outfile).

אתם יכולים להניח שבקבצים infile1 ו infile2 יש מילים ובין כל שתי מילים יש רווח יחיד. נסמן ב n את מספר המילים המופיעות בשני הקבצים יחד. בשתי הרשימות מופיעים המספרים השלמים מ- 0 ועד n-1 כולל כך שכל מספר מופיע ברשימה אחת בדיוק וכל אחת מהרשימות ממויינת מהקטן לגדול.

על הפונקציה ליצור קובץ שיאחד את תוכן שני הקבצים באופן הבא. lst1 מתארת את מיקומי המילים מ infile1 בקובץ המאוחד, ו lst2 מתארת את המיקומים עבור המילים מ infile2. המילה ה i מ infile1 תופיע בקובץ הפלט outfile במקום שמופיע באינדקס i ב lst1 (וכנ"ל לגבי lst2 ו infile2). כלומר אם הרשימה היא lst1=[0,4] ובקובץ יש את שתי מילים "good evening" אז בקובץ המאוחד, המילה good תיהיה המילה הראשונה והמילה evening תיהיה המילה החמישית. שאר המילים בקובץ המאוחד יושלמו ע"פ lst2 ו- infile2 בדיוק באותה החוקיות.

אתם יכולים להניח שהקלט תקין ושניתן לאחד את הקבצים כמתואר בשאלה.

לדוגמא אם תוכן הקובץ infile1 הוא

python is best the world

ותוכן הקובץ infile2 הוא

the course in whole

ושתי הרשימות יהיו

lst1 = [0, 1, 3, 6, 8]
lst2 = [2, 4, 5, 7]

אז תוכן הקובץ outfile יהיה

python is the best course in the whole world

```
def merge_files(infile1, infile2, lst1, lst2, outfile):
```

סעיף ב (14 נקודות)

הניחו כי בתחילת כל תת סעיף בשאלה זו, רצה השורה הבאה:

```
import pandas as pd
```

בסעיף זה נרצה לעזור לירקן לנהל את הסחורה שלו. לפניכם קובץ CSV שמכיל את:

```
Name      Quantity  Sold     Type
Orange         120      30  Fruit
Apple          300      50  Fruit
Cucumber       250     100  Vegetable
Melon          100      22  Fruit
Carrot         400     240  Vegetable
Onion          330     250  Vegetable
```

טבלה זו מייצגת כמה פירות וירקות נמכרו ביום עסקים אחד. בעמודה הראשונה מופיעים שמות של פירות וירקות, בעמודה השנייה מופיעה הכמות ההתחלתית של הפירות והירקות באותו יום ובעמודה השלישית כמה פירות\ירקות נמכרו באותו יום. כל מה שלא נמכר באותו היום נזרק.

נקרא את תוכן הטבלה לתוך משתנה dataframe באופן הבא:

```
df = pd.read_csv('test.csv')
```

אם נדפיס את df נקבל:

```
print(df)
```

```
       Name  Quantity  Sold      Type
0    Orange       120    30     Fruit
1     Apple       300    50     Fruit
2  Cucumber       250   100  Vegetable
3     Melon       100    22     Fruit
4    Carrot       400   240  Vegetable
5     Onion       330   250  Vegetable
```

בסעיפים הבאים עליכם להשתמש בפקודות pandas ובמשתנה df. על הקוד שתממשו לתמוך בטבלאות מהסוג המתואר בשאלה כאשר נוכל להוסיף עוד שורות המייצגות פירות\ירקות. פתרונות שכוללים לולאות יזכו בניקוד חלקי.

1. (5 נקודות) ממשו קוד אשר מדפיס את שם הפרי\ירק ממנו נמכרו הכי הרבה פריטים. לדוגמא, הפעלת הקוד על הטבלה הנתונה ידפיס:

```
Onion
```

2. (4 נקודות) ממשו קוד המוסיף עמודה בשם diff שבה כל ערך מציין את מספר הפריטים שנזרקו לפח מאותו ירק\פרי, כלומר כמה ירקות ופירות לא נמכרו באותו יום.

```
        Name  Quantity  Sold       Type  diff
0     Orange       120    30      Fruit    90
1      Apple       300    50      Fruit   250
2   Cucumber       250   100  Vegetable   150
3      Melon       100    22      Fruit    78
4     Carrot       400   240  Vegetable   160
5      Onion       330   250  Vegetable    80
```

3. (5 נקודות) ממשו קוד אשר מדפיס כמה ירקות נמכרו וכמה פירות נמכרו. אם נריץ את הקוד על הטבלה שלנו, נקבל

```
Type
Fruit        102
Vegetable    590
Name: Sold, dtype: int64
```

סעיף ג

1. (6 נקודות) ממשו את הפונקציה is_condensed(im, ths) אשר מקבלת מערך numpy דו מימדי אשר מייצג תמונה בגווני אפור (im) וערך מספרי בין 0 ל-1 (ths). נסמן את ממוצע התמונה במשתנה x. הפונקציה תחזיר True אם כל ערכי המערך נמצאים בטווח (כולל הקצוות) שבין $(1 - ths) \cdot x$ ל-$(1 + ths) \cdot x$. עליכם לפתור סעיף זה <u>ללא שימוש בלולאות</u> (שימוש בלולאות יזכה אתכם בניקוד חלקי).

<u>דוגמא:</u> עבור המטריצה im עם הערכים הבאים

120	120
107	133

וערך ths=0.5, הפונקציה תחזיר את הערך הבוליאני True. אם נקרא לפונקציה עם im והערך ths=0.1, הפונקציה תחזיר False מפני שהערך 133, למשל, מחוץ לטווח בין $(1 - 0.1) \cdot 120$ ל-$(1 + 0.1) \cdot 120$.

```
def is_condensed(im, ths):
```

2. (8 נקודות) ממשו את הפונקציה special_combine(im1, im2) המקבלת שני מערכי numpy דו מימדיים המייצגים תמונה. עליכם ליצור ולהחזיר מערך numpy דו מימדי חדש שנקרא לו im אשר ישלב את שתי התמונות באופן הבא:
מספר השורות בתמונה החדשה im יהיה שווה למספר השורות ב- im1
מספר העמודות בתמונה החדשה im יהיה שווה למספר העמודות ב- im1 ועוד פעמיים מספר העמודות ב- im2.
עליכם למקם את התמונה im1 במרכז im כך שמשמאל ומימין ל- im1 יופיעו עותקים של התמונה im2. כלומר, הקצה השמאלי העליון של im1 יתחיל במיקום (0, im2.shape[1]). השלמת העותקים של im2 ב-im תתבצע כמתואר בדוגמא הבאה:

im2

100	0
50	200

im1

200	255	22	200
100	100	44	200
33	100	100	255
100	22	100	255

התמונה החדשה תיהיה:

			Im1						
im2	100	0	**200**	**255**	**22**	**200**	100	0	im2
	50	200	**100**	**100**	**44**	**200**	50	200	
im2	100	0	**33**	**100**	**100**	**255**	100	0	im2
	50	200	**100**	**22**	**100**	**255**	50	200	

הניחו כי מספר השורות של im2 מחלק את מספר השורות של im1.

```
def special_combine(im1, im2):
```

מס טופס:_____מספר ת"ז:_____

מסגרת חירום 1

מס טופס:_____מספר ת"ז:_____

מסגרת חירום 2