מסדי נתונים תרגיל 3

חלק ב

שאלה 1

א. חישוב כמות הבלוקים = מספר שורות **חלקי** מספר שורות בבלוק (מספר שורות בבלוק = גודל הבלוק חלקי גודל שורה)

```
>>> block_size = 1024
>>> members_rows = 1000
>>> row_size = 64
>>> rows_in_block = 1024/64
>>> rows_in_block
16.0
>>> members_blocks = members_rows / rows_in_block
>>> members_blocks
62.5
```

תשובה סופית: 63

ב. נוסחה : (גודל הבלוק + גודל תכונת עליה נעשה אינדקס) **חלקי** (גודל מצביע + גודל תכונה עליה נעשה אינדקס)

```
>>> pointer_size = 4
>>> key_size = 4
>>> (block_size+key_size)/(pointer_size+key_size)
128.5
```

תשובה סופית: 128

ג. גובה העץ (צעד 1) + פעולת IO אחת

```
>>> tree_height = math.ceil(math.log(members_rows, 128/2))
>>> tree_height + 1
3
```

תשובה סופית: 3

שאלה 2

א. גובה העץ (צעד 1) + מספר העלים הרלוונטים לפי פילוג אחיד בנתון (צעד 2)

```
>>> birth_rows = members_rows * (10/100)
>>> matching_leaves = math.ceil(birth_rows/(128/2 - 1))
>>> tree_height+matching_leaves
4
```

תשובה סופית: 4

שאלה 3

א. נוסחה : (גודל הבלוק + גודל תכונת עליה נעשה אינדקס) **חלקי** (גודל מצביע + גודל תכונה עליה נעשה אינדקס)

```
>>> pointer_size = 4
>>> key_size = 4
>>> (block_size+key_size)/(pointer_size+key_size)
128.5
```

תשובה סופית : 128

ב. גובה העץ (צעד 1) + תנועה על העלים (צעד 2) + חילוץ תכונה שאינה על האינדקס בפעולות IO כאשר קריאה של כל הבלוקים איננה המינימלית (צעד 3)

```
>>> tree_height = math.ceil(math.log(members_rows, 128/2))
>>> io_leaf = 1
>>> io_block = 1
>>> tree_height+io_leaf+io_block
4
```

תשובה סופית : 4

שאלה 4

א. (גודל הבלוק + גודל תכונת עליה נעשה אינדקס) **חלקי** (גודל מצביע + גודל תכונה עליה נעשה אינדקס)

```
>>> key_size = 10
>>> pointer_size = 4
>>> int((block_size+key_size)/(pointer_size+key_size))
73
```

תשובה סופית :73

ב. גובה העץ (צעד 1) + תנועה על העלים (צעד 2) + קריאה של כל הבלוקים מאחר והינה המינימלית מאשר כמות השורות הרלוונטיות (צעד 3)

```
>>> tree_height = math.ceil(math.log(members_rows, math.ceil(73/2)))
>>> hebrew_rows = members_rows/5
>>> hebrew_leaves = math.ceil(hebrew_rows / (math.ceil(73/2)-1))
>>> tree_height + hebrew_leaves + math.ceil(members_blocks)
71
```

תשובה סופית : 71

ג. נוסחה : (גודל הבלוק + גודל תכונת עליה נעשה אינדקס + גודל תכונה השנייה עליה נעשה האינדקס) **חלקי** (גודל מצביע + גודל תכונה עליה נעשה אינדקס + גודל תכונה השנייה עליה נעשה האינדקס)

```
>>> birth_key_size = 4
>>> language_key_size = 10
>>> key_size = birth_key_size + language_key_size
>>> pointer_size = 4
>>> int((block_size+key_size)/(pointer_size+key_size))
57
```

תשובה סופית : 57

ד. גובה העץ (צעד 1) + תנועה על העלים הרלוונטים (צעד 2) **(אין צורך בצעד שלוש מאחר והתכונית נמצאת כאינדקס)**

```
>>> tree_height = math.ceil(math.log(members_rows, math.ceil(57/2)))
>>> hebrew_leaves = math.ceil(hebrew_rows / (math.ceil(57/2)-1))
>>> tree_height+hebrew_leaves
11
```

תשובה סופית : 11

גודל בלוק הוא 1,024 בייטים.

- בטבלה members יש 1,000 שורות,
- כל שורה תופסת 64 בייטים.
- התכונה birthYear תופסת 4 בייט.
- התכונה uid תופסת 4 בייט.
- התכונה language תופסת 10 בייט.
- מצביע תופס 4 בייט.
- הערכים ב birthYear בטבלה members מתפלגים אחיד בטווח]1900,2000)
- הערכים ב language בטבלה מחולקים ל5 קטגוריות באופן אחיד.

שורות בבלוק: 16
בלוקים לממברס: 63