מבנה נתונים

שאלה 1 : חוליות

אפשר לייצג את הזיכרון במחשב באמצעות רשימה (שרשרת חוליות), כאשר כל איבר ברשימה מכיל את גודל קטע הזיכרון (בבתים) ומציין אם הקטע פנוי או תפוס. כל מקטע מיוצג באמצעות המחלקה Data.

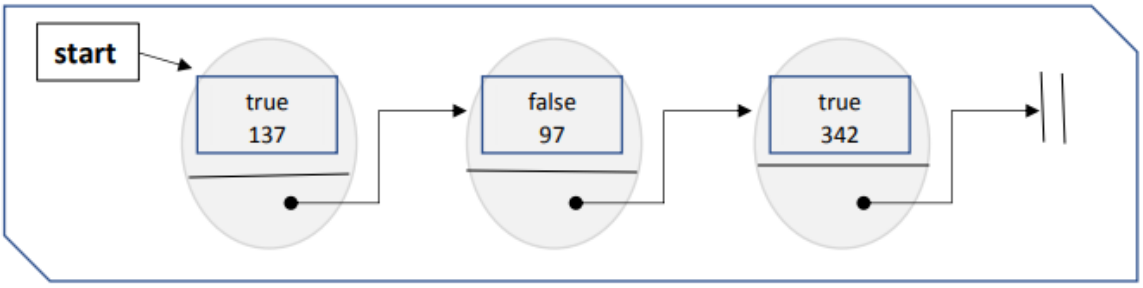
```
public class Data{
private bool      free;
private int size;
//constructor
public Data (int size) {
this.free = true;
this.size = size;
 }
public bool       isFree(){ return free; }
public int getSize() { return size; }
public void setFree(bool       free){this.free = free; }
public void setSize(int size){ this.size = size;}

}
```

הזיכרון כולו מיוצג באמצעות האובייקט הבא:

```
public class Memory {
private Node<Data> start;
public Memory(int totalSize) {
 this.start = new Node<Data>(new Data(totalSize));
}
}
```

להלן דוגמה לזיכרון שיש בו מקטע פנוי בגודל 137 ,אחריו מקטע תפוס בגודל 97 ולבסוף מקטע פנוי בגודל 342:



<u>הערה:</u> לא ייתכן שיהיו שני איברים סמוכים במצב true .אבל זה כן ייתכן לגבי מצב false.

זיכרון מחשב נמצא ב"מצב מסוכן" (State Dangerous) אם כמות הזיכרון הפנוי יורדת מתחת 10% מכמות זיכרון הכללי.

א. כתבו פעולה פנימית במחלקה Memory , הבודקת את מצב הזיכרון והמחזירה true , אם הוא ב"מצב מסוכן", ולא הפעולה מחזירה false.

כאשר המעבד צריך להקצות זיכרון בגודל מסוים (num) הוא יכול להשתמש באלגוריתם First Fit.האלגוריתם מחפש את מקטע הזיכרון הפנוי הראשון שיכול להכיל את num (כלומר, שגודלו הוא לפחות num) ומקצה לו מקום בזיכרון וקובע שמקטע הזיכרון במצב תפוס (false) .

אם הזיכרון נמצא ב"מצב מסוכן" האלגוריתם אינו מבצע דבר.

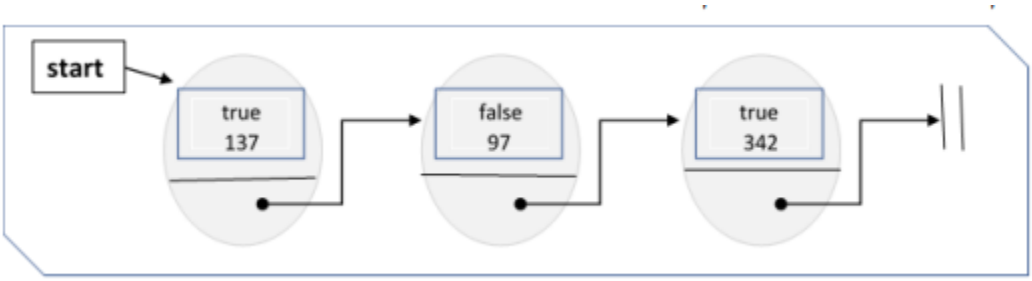
ב. כתבו במחלקה Memory פעולה המממשת את האלגוריתם. כותרת הפעולה:

bool firstFit(int num)

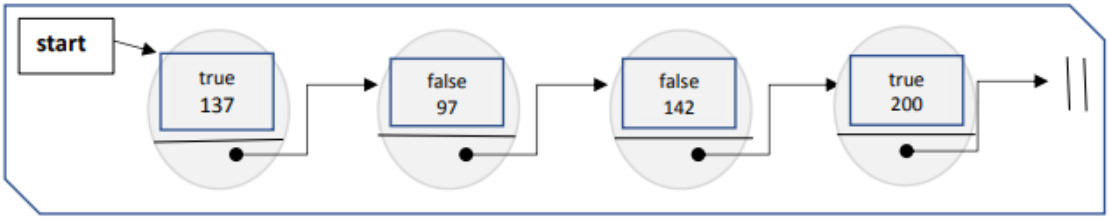
פעולה זו מקבלת כפרמטר את num שהוא גודל הזיכרון הנדרש, מאתרת את המקטע הראשון הפנוי שגדול או שווה ל num , משנה את הרשימה באמצעות הכנסת חוליה המייצגת מקטע תפוס בגודל num ומעדכנת את גודל המקטע הפנוי. הפעולה מחזירה true אם נמצא מקום כזה ו false -אם לא נמצא מקום. אם הזיכרון נמצא ב"מצב מסוכן" הפעולה מחזירה false.

לדוגמה:

אם לפני הזימון של הפעולה מצב הזיכרון הוא



אז אחרי הזימון firstFit(142) מצב הזיכרון ישתנה למצב הבא והפעולה תחזיר true.



שאלה 2 : חוליות

חברת הובלות **TrickTruck** מעוניינת לעקוב אחרי תצרוכת הדלק של כל אחד מכלי הרכב שהיא מפעילה, במשך התקופה של חודש יוני (30 יום).

כל נהג מתדלק מביא לתחנה חשבונית המכילה את הפרטים הבאים:

- יום התדלוק (מספר שלם בין 1-30).
- מספר רכב (מספר שלם).
- שם הנהג המתדלק (מחרוזת).
- כמות הדלק שתודלקה בתדלוק זה (מספר ממשי).

<u>רכב יכול להיות מתודלק כמה פעמים ביום או לא מתודלק כלל ביום מסוים.</u>

במחלקת חשבונית Payment הוגדרו פעולות הבאות:

- פעולה בונה היוצרת חשבונית הכוללת את שם הנהג name את מספר הרכב num ואת כמות הדלק שתידלק fuel.

Payment (String name, int num, double fuel)

- פעולות get ו – set -לכל תכונה והפעולה toString המחזירה מחרוזת המתארת את מצב החשבונית.

המידע עבור כל התדלוקים בחודש אפריל נשמר במחלקה TrickTruck במערך חד-ממדי payments של רשימות מסוג חשבונית Payment . כל תא במערך מייצג יום תדלוק. שתי תכונות נוספות הן רשימת הנהגים drivers ורשימת הרכבים cars.

```
class TrickTrack
{
private Node<String> drivers;
private Node<Integer> cars;
private Node<Payment>[] payments;
….
```

במחלקה הוגדרו פעולה בונה, פעולות set/get.

במחלקה הוגדו שלוש פעולות בוליאניות נוספות:

- existDriver(String name) - הפעולה מחזירה true אם הנהג name עבד בחודש יוני, ואם לא הפעולה מחזירה false.
- existCar(int num) - הפעולה מחזירה true אם הרכב num עבד בחודש יוני, ואם לא הפעולה מחזירה false .
- worked(String name , int day) - הפעולה מחזירה true אם נהג name הגיש קבלת תשלום ביום day, ואם לא, הפעולה תחזיר false.

א. כתבו פעולה במחלקה TrickTruck בשם addPayment שתקבל כפרמטר את היום בחודש day , מספר הרכב num , את שם הנהג name ואת כמות הדלק fuel שתדלק . אם כל הפרמטרים שהפעולה מקבלת הם תקינים, הפעולה מוסיפה חשבונית חדשה לרשימה המתאימה במערך payments ומחזירה true, ואם לא הפעולה מחזירה, false ולא מבצעת דבר.

ב. כתבו במחלקה TrickTruck את הפעולה totalFuel(int num) המקבלת מספר רכב ומחזירה את כמות הדלק הכללית שתדלקו את הרכב.

ג. כתבו במחלקה TrickTruck את הפעולה printWorkDays() המדפיסה עבור כל נהג את מספר הימים שעבורם הוא הציג קבלות.