

מת"מ - 234124

תרגיל בית 1 – יבש

מגישים:

איוון גלאס

ivan@campus.technion.ac.il

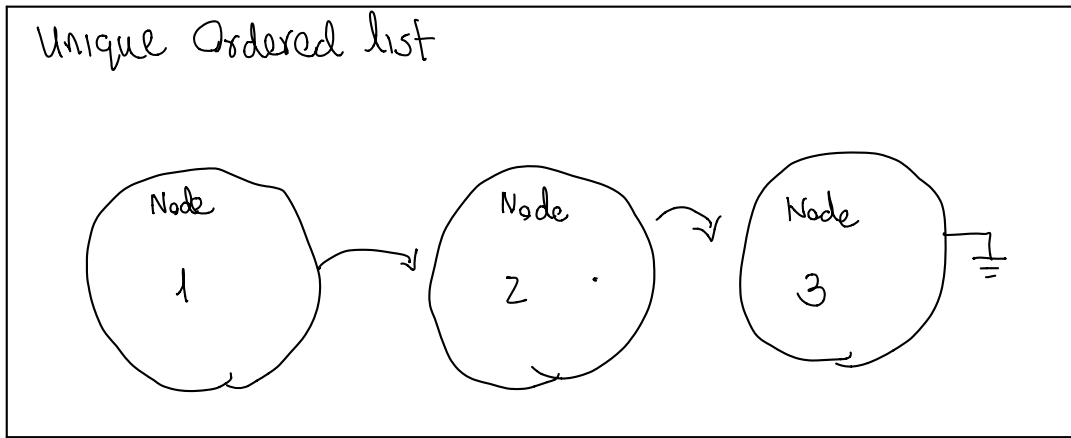
320549124

בר רוח

bar.raveh@campus.technion.ac.il

208950238

תאריך הגשה: 2018-12-13



3.2

Preference

```

int candidateId
int priority
  
```

Citizen - preference
מספר לזכר זכור
למספרים המיועדים.

Citizen #include preference

```

int citizenId
int age
int yearsOfEducation
char* name
bool isCandidate

UniqueOrderedList preferences
  
```

Citizen - נקודת זכור שמירת פרטי כל תושב
עבור City. מנתבם למאפיינים שהתבדקו
לממש מניסוח שמירת המספרים שכלי הוא
נתפסם מוכנה, ורשומה ייחודית למחשבת
המכונה את הנדסות התעסוקה (השימור של
preference)

Node

```

Element data;
Node next
copyElements copyData
freeElements freeData
elementsEquals equalsData
  
```

המכונה של Node
מקומות יחידה ולא ממויינת כש-אזכור
פנימי.

UniqueOrderedList #include Node

```

Node head
Node* iterator
copyElements copyNode
freeElements freeNode
elementsEquals equalsNode
elementsGreaterThan greaterNode
int size
  
```

UniqueOrderedList - מנתבם זכור השימור
המקומות של head במוסדותיים אחרים
המכונה זכור אחר פנימי

```

City
#include citizen
#include candidate

int cityId
char* name

UniqueOrderedList citizens

UniqueOrderedList candidates

```

```

Candidate
int candidateId
char* name

```

candidate - מספר סדרות מידע
 city - מספר מידע
 MtmElections - city

```

MtmElections
#include city
#include candidate

UniqueOrderedList cities

```

city - מספר מידע
 MtmElections - מספר מידע
 (city) - מספר מידע
 (candidate) - מספר מידע

MtmElections - מספר מידע
 cities - מספר מידע
 cities - מספר מידע

3.2

```

5 #include <stdbool.h>
6 #include <stddef.h>
7
8 typedef struct node_t* Node;
9
10 struct node_t{
11     int id;
12     Node next;
13 };
14 //this function works for both question A and B
15 // using Floyd circle finding algorithm
16 bool isCyclic(Node head){
17     if(head == NULL){
18         return true;
19     }
20     Node slowPtr = head;
21     if(!head->next){
22         return false;
23     }
24     Node fastPtr = head->next->next;
25     while(slowPtr && fastPtr && fastPtr->next){
26         if(slowPtr->id == fastPtr->id){
27             return true;
28         }
29         slowPtr = slowPtr->next;
30         fastPtr = fastPtr->next->next;
31     }
32     return false;
33 }

```