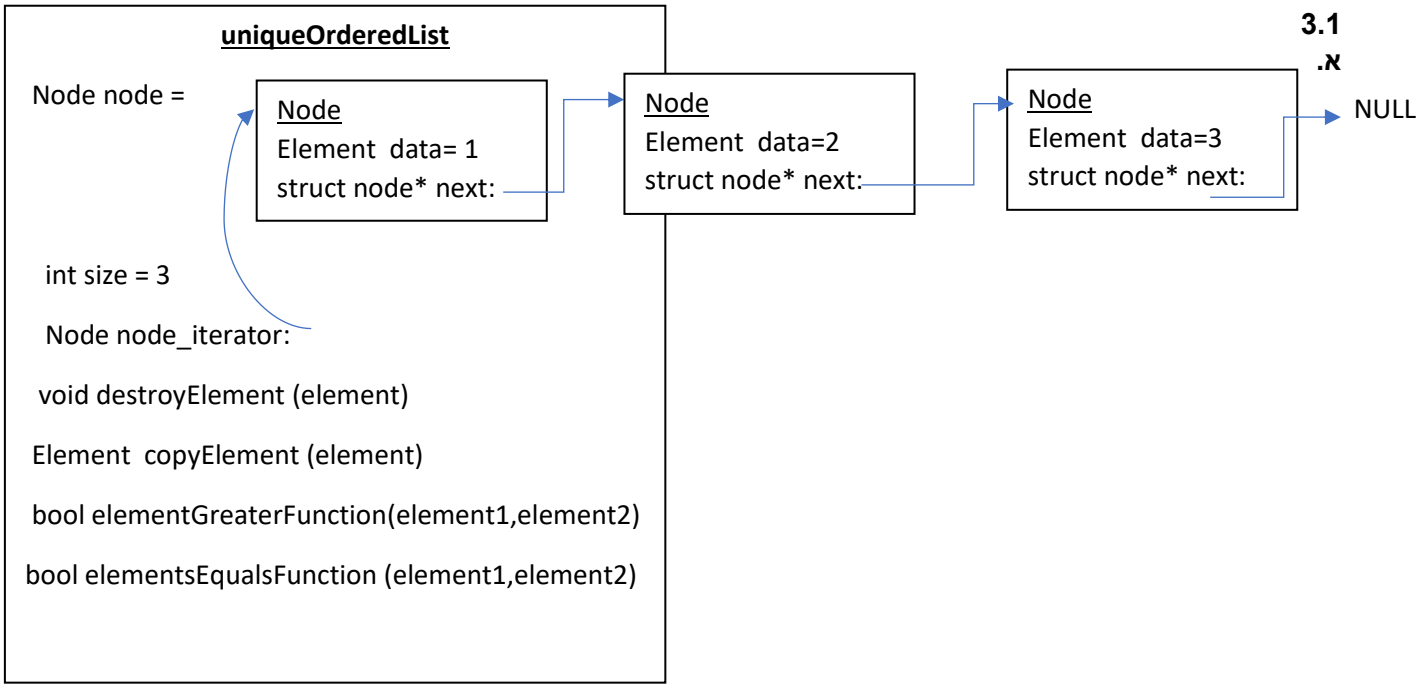
מבוא לתכנות מערכות - 234124
תרגיל בית 1- חלק יבש

שם: נדב אביוב
ת.ז: 206023772

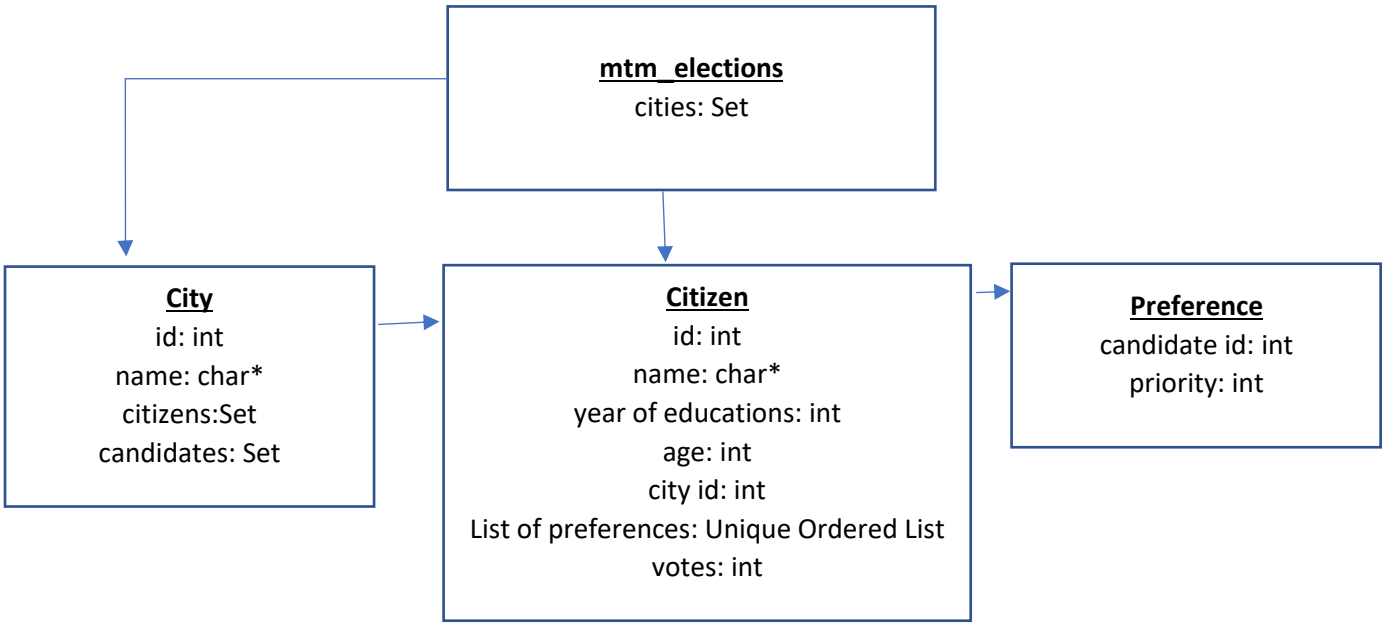
שם: מאי פלסטר
ת.ז: 207683418

3.1

א.



ב.



ג. הציור בסעיף א' מתאר את המבנה של הרשימה הייחודית הממוינת כאשר היא מכילה 3 איברים (1,2,3).
בתוך המבנה של הרשימה הממוינת קיים מבנה של node שהוא האיבר הראשון ברשימה. משמעות החיצים בציור זה היא תיאור מצביעים, כלומר, הnext של כל איבר ממצביע לNode הבא.
כמו כן, האיטרטור (node_iterator) במצב ברירת המחדל שלו מצביע לאיבר הראשון ברשימה.
במצב הנתון הקיים, גודל הרשימה שווה ל-3, וגם נתון זה מופיע במבנה הרשימה.
כל הפונקציות המגדירות את הרשימה גם כן שמורות בתוך מבנה הרשימה

הציור בסעיף ב' מכיל את כל האובייקטים שהשתמשנו בהם במימוש מערכת הבחירות. כל מלבן מייצג adt במימוש שלנו. החיצים מתארים איזה adt מכירים אחד את השני. (למשל mtm_elections מכיר את city ואת citizen, וcity מכיר את citizen).
בתוך כל מלבן מופיעים הנתונים שהגדירו כל מבנה והטיפוס שלהם, או מבנה הנתונים בהם שמרנו אותם.
בתוך הadt הראשי, mtm_elections, שמרנו את הערים בתוך סט.
בתוך city שמרנו (מלבד הנתונים של שם העיר ומזהה העיר) בתוך סט את כל תושבי העיר ואת כל מועמדי העיר (שהם **כולם** מבנה מסוג citizen). בחרנו לאחד את המבנה של האזרחים והמועמדים בשל הדמיון הרב ביניהם (כל הנתונים שמאפיינים תושב מאפיינים אותו גם בתור מועמד, ומועמד חייב להיות גם אזרח). במקום להפריד את המבנים, בחרנו להוסיף למבנה האזרח נתונים שיקבלו אפיון רק אם התושב הוא גם אזרח. עם זאת, בחרנו לשמור את המועמדים של העיר בסט נפרד.
בתוך citizen, שמרנו את כל הנתונים שהתבקשנו (שם, גיל, השכלה, וכדומה). בנוסף, שמרנו את כל ההעדפות שלו בתוך רשימה ייחודית ממוינת. קריטריון ההשוואה של הרשימה היה מזהה המועמד (כך הובטח שלא תהיה העדפה לאותו מועמד). קריטריון הסדר היה על פי ההעדפה של המועמד (כך האיבר הראשון ברשימה היה תמיד המועמד המועדף ביותר). בנוסף, הוספנו בתוך המבנה של האזרח שדה של קולות, שהתחשבנו בו רק כאשר האזרח הוא גם מועמד, והשתמשנו בו לצורך ספירת הקולות וחישוב המנצח.
בתוך Preference שמרנו את מזהה המועמד ואת ההעדפה עבורו.

3.2

א+ב

```
#include <stdlib.h>
#include <stdbool.h>
#include "set.h"

typedef struct node_t* Node;
struct node_t {
   int id;
   Node next;
};

/**
 *
 * @param pointer_to_address1 first node pointer
 * @param pointer_to_address2 second node pointer
 * @return
*    0 if they're equal (point to same address)
*    -1 if the addresses are not equal or pointing to NULL.
 */
int addressCompare (Node pointer_to_address1, Node pointer_to_address2){
   if (!pointer_to_address1 || !pointer_to_address2){
      return -1;
   }
   if (pointer_to_address1 == pointer_to_address2){
      return 0;
   }
   return -1;
}
```

```
/**
 *
 * @param pointer_to_address: the node pointer we want to free
 */
void addressFree (Node pointer_to_address){
//first check if address is not NULL
   if (!pointer_to_address){
      return;
   }
   free(pointer_to_address);
}

/**
 *
 * @param pointer_to_address we want to copy
 * @return
 *  return the address pointer (if pointer_to_address if NULL the function returns NULL)
 */
Node addressCopy (Node pointer_to_address){
   return pointer_to_address;
}

/**
 *
 * @param head pointer to the first node in the list
 * @return
 * true if the id cycle in the list
 * false otherwise (NULL argument, memory error, or no cycle in the list)
 */

bool isCyclic (Node head) {
   if (!head){
      return false;
   }
   Set address_set = setCreate((copySetElements)&addressCopy,
         (freeSetElements)&addressFree, (compareSetElements)&addressCompare);
   if (!address_set){
      return false; //in case of memory error
   }
   Node temp_node = head;
   while (temp_node->next != NULL){
      if (setAdd(address_set,temp_node) == SET_ITEM_ALREADY_EXISTS){
         setDestroy(address_set);
         return true;
      }
      else {
         temp_node = temp_node->next;
      }
   }
   setDestroy(address_set);
```

```
    return false;
}
```

הסבר:

יצרנו סט עזר כדי לשמור בו את כל הכתובות שאליו מצביעים כל האיברים.
עבור כל האיברים ברשימה, שומרים את הכתובת אליו מצביע שדה הnext בתוך הסט החדש. אם מנסים להכניס כתובת שכבר קיימת בסט, נקבל הודעה שגיאה מהסט, שכן האיבר כבר קיים. במקרה זה, אנחנו יודעים שכבר יש איבר שהצביע לכתובת זו. זה אומר שהגענו שוב לאותו איבר ברשימה, והמשמעות היא שקיים מעגל ברשימה.
ביציאה מהלולאה, אם עברנו על כל האיברים עד שהגענו לNULL, המשמעות היא שאין מעגל ברשימה.
הפיתרון מתאים גם לסעיף א' וגם לסעיף ב', כיוון שהוא מתחשב בכתובת של האיברים ברשימה ולא בערך שלהם.