חלק 1- גיבוב אוניברסלי:

הוכיחו או הפריכו על כל אחת מהדוגמאות: **המשפחה H** היא משפחה אוניברסלית.

1. עבור m גודל טבלת הגיבוב, עבור p=m מספר ראשוני, ו- k מספר טבעי חיובי כלשהו בטווח שבין 0 ל-p-1. עולם המפתחות U מכיל מספרים בטווח שבין 0 ל-p-1. נגדיר:

יהי b, c מספרים בטווח 0 עד p-1, ויהי a מספר שלילי בטווח 0 עד (p-1)-.
נגדיר:

$$h_{a,b,c}(k) = (ak^2 + bk + c) \bmod p)$$

נגדיר את H באופן הבא:

$$H = \{h_{a,b,c} | a \in \{0, -1, .., -(p-1)\},\ b, c \in \{0, 1, ..., p-1\}\}$$
$$= \{h_{0,0,0},\ h_{-1,0,1},\ h_{-1,1,0}, ..., h_{-(p-1),p-1,p-1}\}$$

כלומר בחירה אקראית של h מהמשפחה H היא בחירה אקראית של a,b,c באופן בלתי תלוי.

2. נסמן:

$$U = \{000, 001, 002, ..., 999\}$$

כלומר אוסף כל המספרים בין 0 ל999 מרופדים באפסים מובילים כך שהם בני 3 ספרות. יהיה m=10 (גודל טבלת הגיבוב) ויהי a מספר שלם בין 1 ל9. נסמן ב$h_a(x)$ את הספרה הימנית ביותר של $a * x$
כלומר:

$$h_2(123) = 6$$

כי 6 היא הספרה הימנית ביותר של 24**6** = 2*123.
נגדיר את H משפחת הפונקציות הבאה: $H = \{h_1, h_2, ..., h_9\}$

חלק 2: גיבוב k-אוניברסלי וopen addressing

1. גיבוב k-אוניברסלי:

יהי H משפחת פונקציות גיבוב שממפה $U \to [m]$. הוכיחו או הפכיחו את הטענות הבאות:

א. אם H היא 1-אוניברסלית אזי היא גם 2-אוניברסלית

ב. אם H היא אוניברסלית אזי היא גם 2-אוניברסלית.

2. גיבוב באמצעות open-addressing:

יהי $c_1 = 1, c_2 = 2$, $h_2(x) = 1 + \left(\left\lfloor \frac{x}{m} \right\rfloor mod\ m\right)$, $h_1(x) = x\ mod\ m$, $m = 11$. **הכניסו** את הערכים הבאים: 57,76,63,77,54,31,23,74 (משמאל לימין) לטבלת גיבוב מגודל m כאשר התנגשויות מנוהלות לפי 3 השיטות שראינו. הפונקציה $h(x,i)$ מייצגת את הניסיון הi לגבב את הערך x:

א. Linear probing: $h(x,i) = (h_1(x) + i)\ mod\ m$

ב. Quadratic probing: $h(x,i) = (h_1(x) + c_1 i + c_2 i^2)\ mod\ m$

ג. Double hashing: $h(x,i) = (h_1(x) + i h_2(x))\ mod\ m$

בנוסף, ספרו את מספר ההתנגשויות שקרו במהלך השימוש בכל שיטה. האם זה צירוף מקרים?

הגישו רק את הטבלה הבאה מלאה:

	0	1	2	3	4	5	6	7	8	9	10	# collisions
Linear probing												
Quadratic Probing												
Double Hashing												

חלק 3- שימושים של hash-tables:

1. יהי A מערך כלשהו, אזי עבור a ששייך לA נגדיר את התדירות של a להיות מספר הפעמים שa מופיע בA. הציעו ותארו אלגוריתם PrintFrequencies(A) שמקבל מערך כלשהו A בגודל n ומדפיס זוגות (a, i_a) לכל a ששייך לA כך שi_a מוגדר להיות התדירות של a בA. האלגוריתם שאתם מתארים צריך לרוץ בזמן ריצה $O(n)$בתוחלת. הסבירו את זמן הריצה של האלגוריתם.

2. הציעו מבנה נתונים עם הפונקציות הבאות:
   a. insert(x) :
      הכנסת איבר חדש למבנה הנתונים בזמן O(1) בתוחלת.
   b. find(x) :
      מציאת איבר במבנה הנתונים בזמן O(1) בתוחלת.
   c. delete_oldest() :
      מחיקת האיבר האחרון שהוכנס לתוך מבנה הנתונים בזמן ריצה $O(1)$ בתוחלת.

   תארו כיצד ניתן לבנות את מבנה הנתונים הזה, וכיצד כל אחת מהפונקציות תעבוד.

3. הציעו מבנה נתונים שיכיל n איברים כאשר לכל איבר יש key ייחודי כלשהו ויש value מסוים שמאותחל להיות 0. במבנה הנתונים ניתן לעשות את הפעולות הבאות:

   א.
      i. find(key): החזרת ערך
         החזר את ערך ה-value אשר מתאים לkey. בזמן ריצה O(1) בתוחלת
      ii. increment(key): העלאת ערך
         העלאת ערך ה-value שמתאים לkey ב1 בזמן ריצה O(1) בתוחלת
      iii. set(key, new_value): שינוי ערך
         שינוי הערך ששמור תחת key ל-value חדש בזמן ריצה O(1) בתוחלת

   זמן הבניה של מבנה הנתונים (הכנסת המפתחות עם הערכים 0) צריך להיות O(n) בתוחלת.

   ב. כעת, מבנה הנתונים צריך לאפשר את הפעולה של increment_all() כלומר העלאת כל הערכים ב1. הפעולה צריכה לרוץ בזמן ריצה O(1) (במקרה הגרוע). הסבירו כיצד תמומש הפעולה וכיצד <u>ישתנו הפונקציות הקודמות</u>. ניתן להשתמש בשטח בגודל O(1) בנוסף לזה שתואר בסעיף הקודם.

   ג. כעת, במקום הפעולה של סעיף ב' מבנה הנתונים יאפשר את הפעולה set_all(new_value) כלומר לשנות את הערכים של כל האיברים לnew_value. על הפעולה לרוץ בזמן ריצה O(1) (במקרה הגרוע). הסבירו את מימוש הפעולה, <u>וגם כיצד ישתנו הפונקציות הקודמות</u>. ניתן להשתמש בשטח בגודל O(n) בנוסף לזה שתואר בסעיף א'.