



במבחן שלפניכם יופיעו 4 שאלות, בהן עליכם לכתוב אלגוריתמים. בכל שאלה כתבו את האלגוריתם היעיל ביותר שאתם חושבים עליו. הסבירו עבור כל שאלה את האבחנות שהובילו אתכם אל האלגוריתם, את דרך פעולתו ומדוע הוא נכון. אין צורך בהוכחה מתמטית פורמלית, רק בהסבר ברור.

את האלגוריתם עצמו אתם יכולים לכתוב בפסאודוקוד, או בשפת תכנות C, Java, C# או C++. בכתיבת פסאודו-קוד אנחנו לא בודקים את חוקי הכתיבה, אלא את הלוגיקה של האלגוריתם עצמו. אתם רק צריכים לכתוב אותו באופן ברור כדי שנבין את הפקודות, לא חייבים להצמד לדרך פורמלית כלשהי (כמו מספור שורות או בחירת מילים ספציפית). הפסאודו-קוד בא להקל עליכם, כדי שלא תצטרכו להתעסק בתחביר של שפות התכנות, ולא להקשות!

אנחנו יכולים גם לבדוק פתרונות בשפת פייתון, אבל לא ממליצים לכם לכתוב בה, מכיוון שקשה לאמוד את היעילות עם מבני הנתונים המורכבים שבה. אם בכל זאת בחרתם לכתוב את האלגוריתמים שלכם בפייתון, אנא רשמו קוד עם פקודות קריאות ופשוטות יחסית, ולא עם one-liners מופצצים.

המבחן בודק את יכולת פתרון הבעיות שלכם, ולא את הידע באלגוריתמים מוכרים. אין צורך להסביר כיצד אתם מממשים אלגוריתמים מוכרים כמו החלפת משתנים, מיון או הפיכת מערך וכו'. ניתן לרשום "החלף בין x ל y", "מיין את המערך A בסדר עולה/יורד", "הפוך את המערך A וכו' בפסאודו-קוד".

בהצלחה!

The following exam contains 4 algorithmic questions. In every question write the most efficient algorithm you can think of.

Explain the insights that led you to the algorithm, and how and why it works. There is no need for a formal mathematical proof, just a clear explanation.

You can describe your algorithms by pseudo-code, or by a program language like C#, Java, C or C++.

We don't care about the formal syntax when you write the pseudo-code (like rows numbering, or specific choice of words).

We accept pseudo-code to make it easier for you to describe the algorithm without dealing with the syntax of programming languages!

We can also accept solutions in python, but we don't recommend you use it, because it is difficult to estimate the efficiency of your code with its complex data structures. If you choose to write your solutions in python anyway, please write your code with clear and simple commands, e.g. don't use some crazy one-liners.

The exam tests your problem-solving ability, not your knowledge in known algorithms. There is no need to explain how you implement known algorithms like swapping variables, sorting or reversing an array, etc. You can write "swap x and y", "sort the array A in ascending / descending order", "reverse the array A" etc. in your pseudo-code.

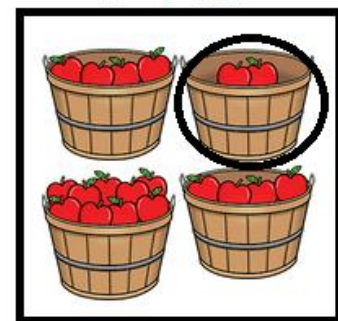
Good luck!



מארזים טעימים

לגאלה אוסף של סלים עם תפוחים, הסלים ממוספרים מ-1 עד n . בכל סל יש מספר תפוחים חיובי כלשהו. גאלה יכולה להרכיב מהסלים הללו מארזים. מארז מורכב מאוסף של סל אחד או יותר (לא בהכרח עם מספרים רצופים). מארז נחשב טעים, אם מספר התפוחים בסל עם הכי מעט תפוחים במארז זה, כפול מספר הסלים בו, גדול מ- x (x הוא מספר שלם נתון). גאלה רוצה להרכיב כמה שיותר מארזים טעימים! (וכמובן שכל סל יכול להיות לכל היותר במארז אחד) עזרו לגאלה, וכתבו עבודה אלגוריתם, שמקבל את x (מספר שלם כלשהו), את n (מספר הסלים) ומערך בשם `apples_arr` באורך n כך שבכל תא במערך ימצא מספר התפוחים בסל המתאים. על האלגוריתם שלכם לחשב את מספר המארזים הטעימים הגדול ביותר שגאלה יכולה להרכיב!

$$2*4 > x?$$



Tasty packages

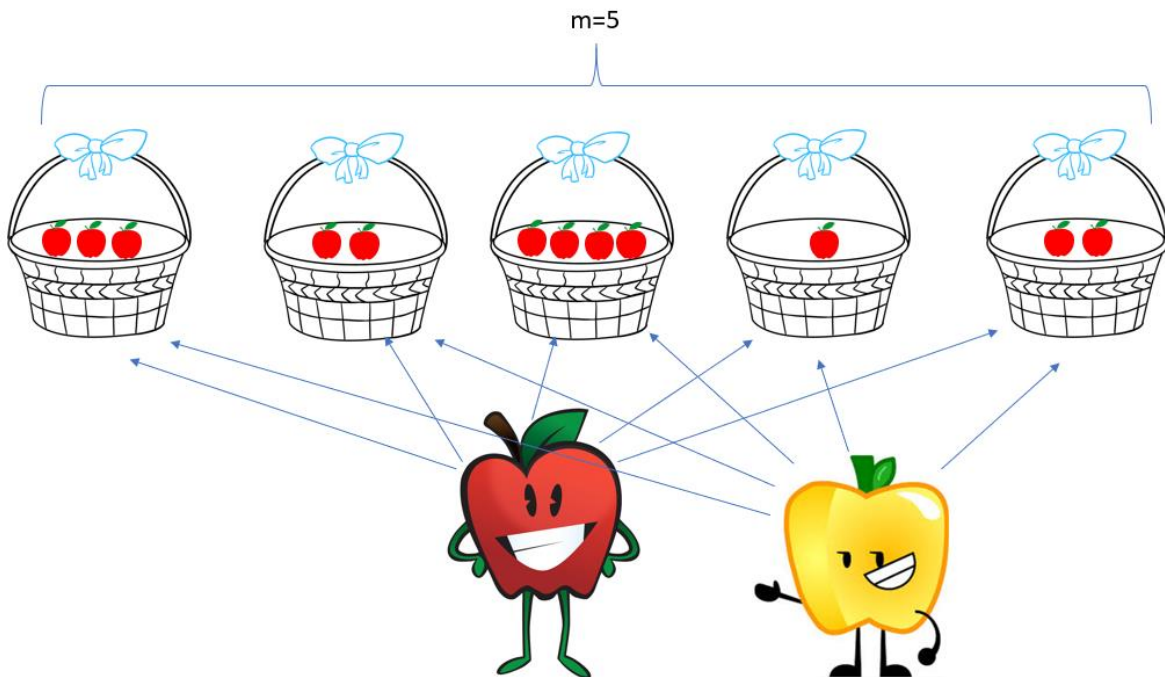
Gala has a collection of baskets with apples. The baskets are labeled from 1 to n . Each basket contains a positive number of apples. Gala can make packages out of these baskets. A package is made of one or more baskets (not necessarily with continues labels). A package is considered tasty, if the number of apples in the basket with the least (minimum) number of apples inside this package, multiplied by the number of baskets in the package, is larger than x (a given integer number). Gala wants to make as many tasty packages as possible! (and of course, every basket can be in no more than one package) Help Gala, and write an algorithm for her, that gets x (an integer number), n (the number of baskets), and an array named `apples_arr` of length n —where each element contains the number of apples in the respective basket. Your algorithm should compute the largest possible number of tasty packages Gala can make!

תפוחונים

האחים רד וגולדן משחקים במשחק התפוחונים. במשחק יש m סלים עם n_1, n_2, \dots, n_m תפוחונים בכל סל בהתאמה (בסל הראשון יש n_1 תפוחונים, בסל השני יש n_2 תפוחונים וכו'). 2 השחקנים רואים כמה תפוחים יש בכל אחד מהסלים. בכל תור השחקן שזהו תורו בוחר את אחד הסלים ואוכל ממנו k תפוחונים (k לא חייב להיות ערך קבוע, השחקן בוחר אותו כל תור בהתאם למגבלות בסעיפים). השחקן שבהגיע תורו אין לו מהלך אפשרי (כלומר אין באפשרותו לאכול k תפוחונים מאחד הסלים ועדיין לעמוד במגבלות על k), מפסיד במשחק. רד מתחיל במשחק.



- בהנחה שרד וגולדן הם שחקנים שלא עושים טעויות ומשחקים תמיד את האסטרטגיה הטובה ביותר, עבור כל אחד מהסעיפים הבאים, כתבו אלגוריתם שמקבל את מספר התפוחונים בכל סל - n_1, n_2, \dots, n_m (מערך n_arr בגודל m), ויקבע האם רד (השחקן שהתחיל ראשון) ינצח במשחק:
- m כללי, k חייב להיות אי זוגי (כלומר בכל תור כל שחקן יכול לקחת כל מספר אי זוגי של תפוחונים מאחד הסלים).
 - $m=2$, k חייב להיות חזקה של 2 (2^i כש- $i \geq 0$ שלם). כלומר יש 2 סלים, ובכל תור כל שחקן יכול לקחת 1, 2, 4, 8 או כל חזקה של 2 תפוחונים מאחד הסלים.
 - m כללי, k חייב להיות חזקה של 2.



Mini apples

Red and Golden play the mini apples game.

In this game there are m baskets with n_1, n_2, \dots, n_m little apples in each of them, respectively (i.e. the first basket contains n_1 little apples, the second pile contains n_2 little apples, etc.).

Both players see the number of apples in each basket.

In each turn, the player chooses one of the baskets and eats k apples from it (k doesn't have to be a constant, the player chooses it according to the limits in each subtask). The player who cannot perform a legal move when it's his turn (i.e. cannot eat k apples from any of the baskets, when the limits on k hold), loses the game. Red makes the first move.

Assuming both players make no mistakes and always play by the correct strategy, write an algorithm for any of the following subtasks, that gets the number of apples in each basket - n_1, n_2, \dots, n_m (an array n_arr of length m), and determines whether Red (who makes the first move) wins the game:

- m can be any positive integer number, and k must be odd (i.e. in each turn the player can take any odd number of apples from one of the baskets).



- b. $m=2$, and k must be a power of 2 (2^i where $i \geq 0$), i.e. there are 2 baskets, and in each turn the player can take 1, 2, 4, 8 or any power of 2 apples from one of the baskets.
- c. m can be any positive integer number, and k must be a power of 2.

החנות של גברת פינק

גברת פינק מנהלת חנות תפוחים. הגיעו אליה n זוגות תפוחים, של תפוח אחד ירוק ותפוח אחד אדום באותו המשקל. המשקלים של הזוגות שונים זה מזה (כלומר יש n משקלים שונים לתפוחים בכל צבע). גברת פינק סידרה את התפוחים האדומים בשורה, וסימנה את המקומות שלהם מ-1 עד n . היא רוצה לסדר את התפוחים הירוקים בשורה מתחת לאדומים, כך שמתחת לכל תפוח אדום יהיה תפוח ירוק באותו המקום (לאו דווקא באותו המשקל). כשלקוח מגיע לחנות, עליו למסור לה את מיקומי התפוחים שהוא מעוניין לקנות, ואז גברת פינק מוכרת לו את התפוח האדום והתפוח הירוק מכל אחד מהמקומות שהוא בחר. הלקוח תמיד בוחר לפחות מקום אחד, ואף פעם לא קונה את כל התפוחים בחנות. עזרו לגברת פינק לסדר את התפוחים הירוקים בצורה כזו, שלא משנה איזה מקומות הלקוח יבחר, המשקל הכולל (כלומר סכום המשקלים) של התפוחים הירוקים שהוא יקבל יהיה שונה מהמשקל הכולל של התפוחים האדומים שהוא יקבל.

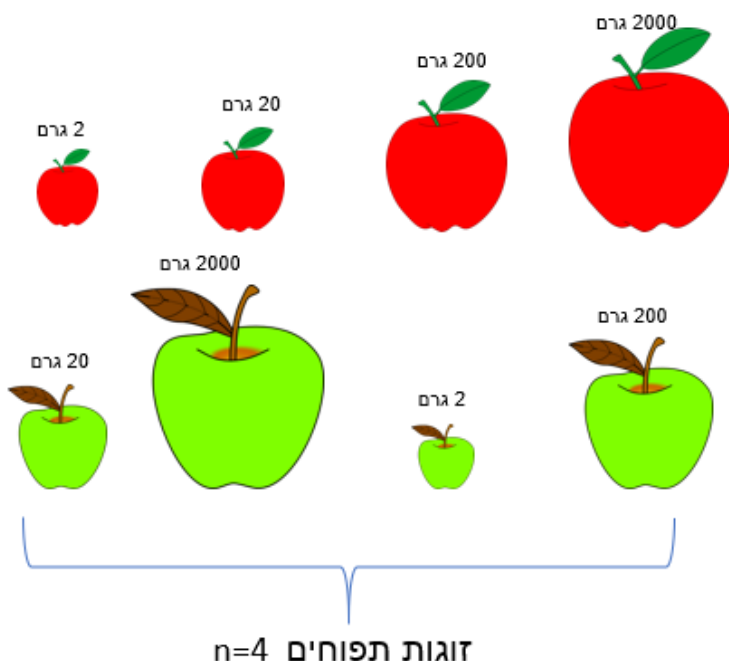
כתבו אלגוריתם, שמקבל את המערך `red_arr` - מערך באורך n שמכיל את המשקל של התפוח האדום בכל מקום בשורה שגברת פינק סידרה, ומדפיס את סידור מתאים של התפוחים הירוקים לפי הסדר (או בונה את המערך `green_arr`).

לדוגמה, אם המערך `red_arr` הוא:

2 20 200 2000

סידור חוקי של התפוחים הירוקים יהיה:

20 2000 2 200





Lady pink's store

Lady Pink runs an apple store. She received n pairs of apples, each consists of one green apple and one red apple of the same weight. The weights of the pairs are distinct from each other (i.e. there are n different weights for the apples of each color).

Lady Pink arranged the red apples in a row, and labeled their positions from 1 to n . She wants to arrange the green apples in a row, just below the red ones, such that below every red apple there will be a green apple in the same position (not necessarily of the same weight). When a customer comes to the store, he has to tell her the positions of the apples he wishes to buy, and then lady Pink sells him the red apple and the green apple from each of the position he has chosen. The customer always chooses at least one position, and never buys all the apples in the store.

Help lady Pink to arrange the apple in a way such that no matter which positions the customer will choose, the total weight (i.e. the sum of the weights) of the green apples he gets will be different than the total weight of the red apples he gets.

Write an algorithm, that gets the array **red_arr** – an array of length n that contains the weight of the red apple in each position in the row lady Pink arranged, and prints a valid arrangement of the green apples – their weights by their order in the row (or builds the array **green_arr**).

For example, if the array **red_arr** is:

2 20 200 2000

A valid arrangement of the green apples with be:

20 2000 2 200

מדף התפוחים

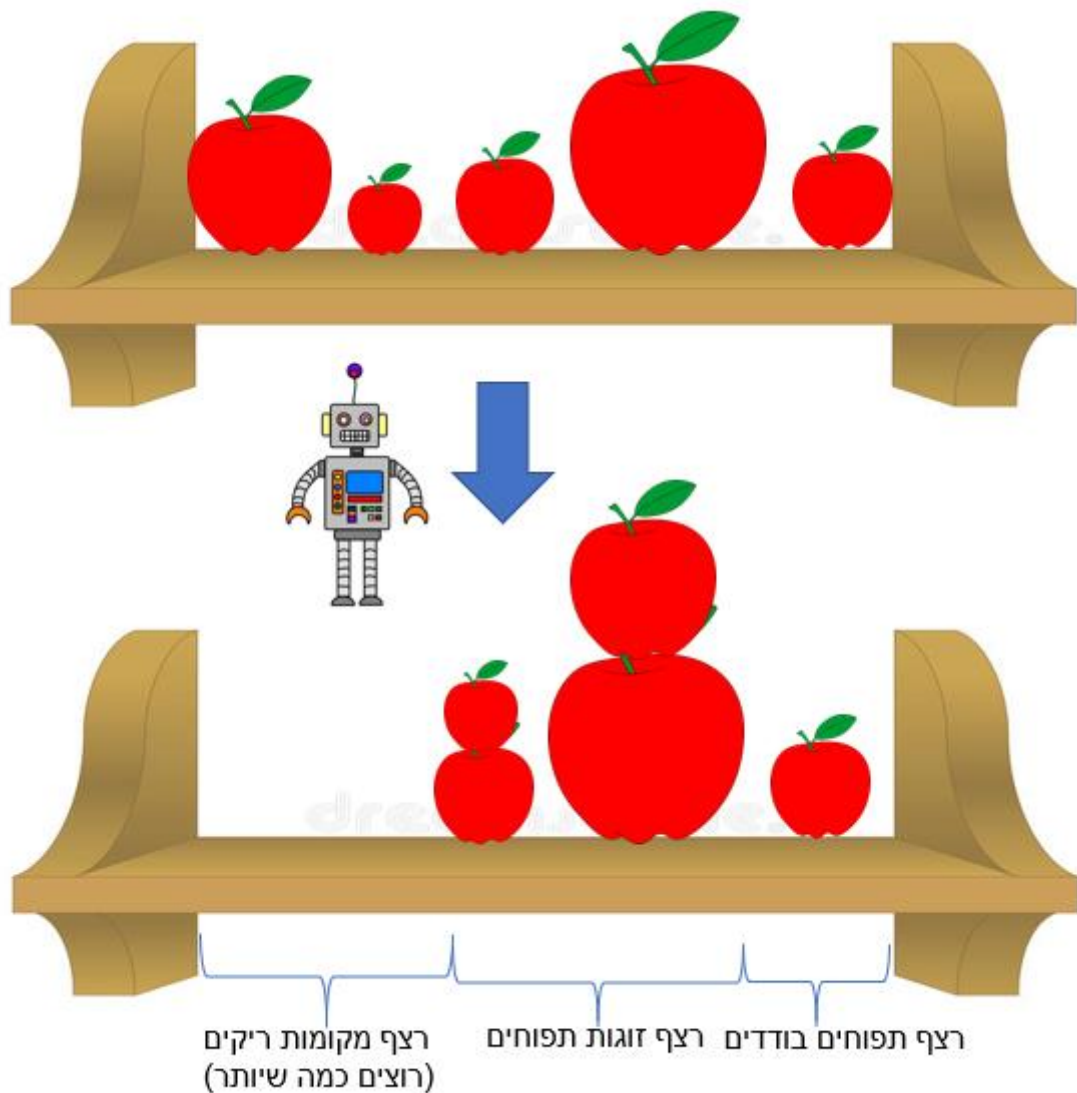
לסבתא סמית מדף עם n תפוחים. על המדף יש n מקומות הממוספרים מ-1 (המקום השמאלי ביותר) עד n (המקום הימני ביותר). כל אחד מהתפוחים של סבתא סמית בגודל שלם כלשהו בין 1 ל- m (יכולים להיות מספר תפוחים באותו הגודל). בתקופת הסגר סבתא סמית פיתחה תחביב חדש של סידור תפוחים על המדף. תחילה, היא שמה בדיוק תפוח אחד בכל אחד מהמקומות. לאחר מכן, היא הזמינה באינטרנט רובוט מיוחד שמסדר תפוחים.

לרוע מזלה, הרובוט שהגיע לא מוצלח במיוחד. הוא יודע רק לקחת תפוח כלשהו שנמצא לבד במקומו, ולשים אותו מעל תפוח כלשהו שגדול ממנו ושנמצא מימינו. הוא לא יכול להזיז תפוח שמאלה, ולא יכול לשים 3 תפוחים ומעלה אחד על השני, רק עד 2. הוא גם לא יכול להניח תפוח במקום ריק, הפעולה היחידה שהוא יכול לעשות היא הפעולה המתוארת. אחלה רובוט.

סבתא סמית רוצה לסדר את התפוחים בסידור יפה. סידור יפה הוא סידור בו בצד הכי שמאלי של המדף יש כמה שיותר מקום פנוי, אחר כך יש זוגות של תפוחים זה על זה ברצף, ואז יש תפוחים בודדים ברצף. במילים אחרות, היא רוצה לשים את K התפוחים השמאליים ביותר על K התפוחים שבדיוק מימינם בסדר כלשהו, כש- K גדול ככל האפשר. סבתא סמית מתעקשת לא לגעת בתפוחים שלא בעזרת הרובוט. עזרו לסבתא סמית, וכתבו עבורה אלגוריתם יעיל שמקבל את n (מספר התפוחים), את m (הגודל המירבי של תפוח), ואת גדלי התפוחים על המדף לפי המיקומים שלהם (במערך **apple_arr**), ומדפיס את K הנ"ל, בהנחה ש (לכל סעיף יכול להתאים אלגוריתם אחר):

א. m גדול משמעותית מ- n

ב. n גדול משמעותית מ- m



The apple shelf

Grandma Smith has a shelf with n apples. On the shelf there are n positions, numbered 1 (the leftmost position) to n (the rightmost position). Each apple has an integer size between 1 to m (some apples might be of the same size). During the lockdown, grandma Smith found a new hobby of arranging the apples on the shelf. First, she put exactly one apple in every position. Then, she ordered a special robot that arranges apples on the internet.

Unfortunately, the robot wasn't turned out to be very successful. It can only take an apple that doesn't share its position on the shelf with another apple and put in on a bigger apple on the right (but not necessarily adjacent). It cannot move an apple to the left, or put more than 2 apples on one another, it can only create "piles" of 2. It also cannot put an apple in an empty position. The only action it can do is as described above.



Grandma Smith wants to rearrange the apples in a beautiful arrangement. A beautiful arrangement is an arrangement in which on the leftmost side of the shelf there is as much free space as possible, to the right of the empty positions there are pairs of apples on one another, and to the right of them there are single apples. In other words, grandma smith wants to put the K leftmost apples on the K apples immediately to their right in some order, when K is as large as possible. Grandma Smith insists not to touch the apples without the robot. Help grandma Smith, and write an efficient algorithm for her that gets n (number of apples and positions on the shelf), m (the maximum size of an apple), and the sizes of the apples on the shelf according to their positions (the array **apple_arr**), and prints the value of K described above, when (each subtask can have a different algorithm):

- a. m is significantly larger than n .
- b. n is significantly larger than m .

בהצלחה!

Good luck!