*אביחי חדאד – 209286665*

*אלעד פישר – 318882800*

*בעיה 1 – הגרלה "עליה וזכיה"*

*קוד*

# Question 1 - getting an number of players and generating

# to each player 10 numbers(in range of 1 - 100), and printing the

# winner(the first player that makes the longest ascending arithmetic sequence).

# importing random for generate\_list() function.

import random

def longestSubarray(a, n):

""" Function longestSubarray gets a list of numbers

and the size of that list.

and returns the size of longest ascending arithmetic sequence

in that list.

"""

# declaring "counters\_list" that will contains all the sizes of the

# ascending arithmetic sequence in the "a".

counters\_list = []

# declaring "counter" with the value 1, that represent the size of

# ascending arithmetic sequence while running the loop.

counter = 1

# running in a loop until before last element in the list

# (because we are checking if the element[i] is smaller than the next element).

for i in range(n - 1):

# checking if the pair of elements(element[i], element[i+1])

# are making an ascending arithmetic sequence.

if a[i] < a[i + 1]:

# if so, incrementing the counter by 1.

counter += 1

# if the pair are not in ascending order, or equal.

else:

# adding the last counter to the "counters\_list".

counters\_list.append(counter)

# resetting the counter to 1 again.

counter = 1

# adding the last counter in the loop to the "counters\_list".

counters\_list.append(counter)

# returning the maximum value in the "counters\_list"

return max(counters\_list)

def generate\_list(ticket\_size):

""" Function generate\_list gets a ticket\_size,

and returns a list with length of ticket\_size

with randomized integer numbers between [1, 99].

"""

# declaring an empty list variable

list = []

# running in a loop "ticket\_size" times.

for i in range(ticket\_size):

# every iteration appending new random number to the list.

list.append(random.randint(1, 99))

# returning the fully randomized list with length of "ticket\_size".

return list

def print\_ticketArray(a, n):

""" Function print\_ticketArray gets a list,

and prints all the elements of the list in the same line

"""

# running a loop of all the elements of the list.

for i in range(n):

# in each iteration printing each element with space afterwards.

print(a[i], end=" ")

def play(number\_of\_players):

""" Function play is the main function of the program,

it gets the "number\_of\_players" from the user and summons the

right functions in order to declare the winner

(the first player that makes the longest ascending arithmetic sequence)

of the lottery game.

"""

# declaring an empty list that will contains

# all the generated list of each player.

all\_players\_list = []

# declaring an empty list that will contains the sizes of

# longest ascending arithmetic sequence of

# each player(according to player index).

list\_of\_longest\_subarray = []

# declaring an constant value for readability.

list\_size = 10

# running in a loop "number\_of\_players" times.

for i in range(number\_of\_players):

# generating a list for each player an appending it

# to the "all\_players\_list".

all\_players\_list.append(generate\_list(list\_size))

# printing the introduction of the

# ticket list(i + 1 for not calling first player -> 0).

print("player", i + 1, "ticket: ", end="")

# printing the ticket list.

print\_ticketArray(all\_players\_list[i], list\_size)

# printing an empty line for separating a player from next player.

print()

# in each iterating calling the function "longest\_subarray"

# for each player and appending the returned

# value the "list\_of\_longest\_subarray".

list\_of\_longest\_subarray.append(

longestSubarray(all\_players\_list[i], list\_size))

# declaring a "winner\_number" for readability in the next print.

# the variable gets the first index that equal to maximum value

# in the "list\_of\_longest\_subarray",

# and that index represent the index of the winner, we added 1 to

# the index because the count of the players starts

# with number 1(not 0 like in a list).

winner\_number = \

list\_of\_longest\_subarray.index(max(list\_of\_longest\_subarray)) + 1

# printing the winner.

print("\*\*\*Player number", winner\_number, "is the winner\*\*\*")

# getting a number of players from the user.

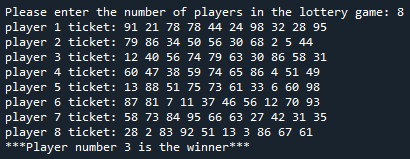
num\_of\_players = \

int(input("Please enter the number of players in the lottery game: "))

# running the function "play" with the "num\_of\_players".

play(num\_of\_players)

*פלט לבעיה 1:*



*בעיה 2 – left\_circular\_shift*

1. תתי-בעיות הקיימות:
2. קליטת אורך רשימה מהמשתמש
3. קליטת מספרים והכנסתם לרשימה
4. הצגת הרשימה
5. שמירת הספרה השמאלית בכל איבר
6. האיבר אשר נשמר מהמספר יצטרך לעבור לתא השמאלי לו לספרת האחדות שלו.
7. לחתוך את הספרה השמאלית מהתא המקורי שלה
8. להעביר את המספר החדש הכולל לתא השמאלי
9. הדפסת הרשימה החדשה

משימות לתתי-בעיות:

לבעיה הראשונה:

1. יצירת ערך שמשתנה על ידי המשתמש שידרוש את אורך הרשימה

לבעיה שנייה:

1. ליצור רשימה ריקה
2. ליצור לולאה שרצה לפי גודל הערך שהוכנס בבעיה הראשונה

בכל ריצה לבקש מהמשתמש קליטה של מספר. לבצע עליו פעולת append לרשימה.

לבעיה השלישית:

1. להדפיס את הרשימה אשר נוצרה עם המשפט שמדגיש שזו אכן הרשימה המקורית

לבעיה הרביעית:

1. לכתוב פונקציה שמקבלת מספר ומחזירה את הספרה הראשונה שלו (במיקום האפס).
2. קריאה לפונקציה לעיל ושמירת הספרה שיוצאת ממנה.
3. אם מדובר באיבר הראשון של הרשימה, אז שמירה במשתנה "מיוחד".

לבעיה החמישית:

1. ליצור פונקציה שמקבלת מספר וספרה ומחברת את הספרה למיקום האחדות במספר.
2. לשלוח את הספרה השמורה מבעיה 4 ואת המספר משמאלו לפונקציה ולעדכן את אותו התא של המספר.

לבעיה שישית:

1. יצירת פונקציה אשר מקבלת מספר ומחזירה מספר אשר לא כולל את הספרה בתא 0 שלה (כלומר, הספרה השמאלית ביותר).

לבעיה שבע:

1. יצירת רשימה חדשה אשר תכיל את האיברים לאחר השינוי
2. ריצה בלולאה על הרשימה המקורית שנוצרה בבעיה 2: עבור כל תא ברשימה, ביצוע בעיות 4 – 6. (עבור התא הראשון דילוג ביצוע בעיה 5)

[בגלל שהתא האחרון עדין "לא קיים"]

1. שמירת המספר החדש ברשימה החדשה לאחר ביצוע הבעיות.
2. לאחר גמר הלולאה, ביצוע בעיה 5 לתא הראשון.

לבעיה שמונה:

1. הדפסת הרשימה עם דגש שזו הרשימה החדשה לאחר הזזה.
2. הצגת תכנון דרך תרשים זרימה:

קליטה מהמשתמש את אורך הרשימה L

לולאה בהתאם לאורך הרשימה: קליטה מהמשתמש אשר מבקשת מספרים שלמים a

בניית רשימה

A = []

לאחר הריצה הL מעבר הלאה

להכניס את המספר הקיים a לרשימה A

יצירת רשימה חדשה

shifted

הדפסת הרשימה המלאה עם דגש שזו הרשימה לפני שינוי

ריצה בלולאה עבור ארוך הרשימה A

שמירת הספרה השמאלית עבור כל איבר

שמירת הספרה השמאלית במשתנה מיוחד בשם

leftest\_of\_first

האם זו הריצה הראשונה?

כן

חיבור הספרה השמאלית שנשמרה לספרת האחדות של התא השמאלי לו

לא

מחיקת הספרה השמאלית בתא והכנסתו לרשימה shifted

הוספת leftest\_of\_first לתא האחרון ברשימה shifted, במיקום ספרת האחדות.

לאחר סיום הלולאה

הדפסת הרשימה shifted עם דגש שמדובר ברשימה לאחר שינוי

1. רשימת משתנים עיקריים:

|  |  |
| --- | --- |
| שם משתנה | מטרה |
| **number** | מייצג את אורך הרשימות |
| **original\_array** | מייצג את הרשימה המקורית |
| **shifted\_list** | מייצג את הרשימה לאחר השינויים |
| **leftest\_digit** | מייצג את הספרה השמאלית של מספר |
| **leftest\_of\_first\_element** | מייצג את הספרה השמאלית של התא הראשון |
| **shifted\_element** | מייצג את המספר לאחר מחיקת הספרה השמאלית |

1. מילוי הטבלה לפי המשימות/תת-בעיות שבחרנו:

|  |  |  |  |
| --- | --- | --- | --- |
| שם הפונקציה | פרמטרים | ערך מוחזר | תיאור |
| deleting\_leftest\_number | number | מחרוזת ללא הספרה השמאלית | מקבלת מספר ומחזירה מחרוזת של אותו המספר ללא הספרה השמאלית |
| get\_leftest\_digit | number | הספרה השמאלית ביותר של המספר | מקבלת מספר ומחזירה את הספרה השמאלית שלו |
| concatenate\_unit\_digit\_to\_number | number, digit | מספר עם ספרת אחדות חדשה של הספרה אשר הוא קיבל | מקבלת מספר וסיפרה ומחזירה לאחר חיבור מחרוזת של המספר עם הספרה, כאשר הספרה ממוקמת כספרת האחדות |
| left\_circular\_shift | list | רשימה חדשה לאחר שינוי | מקבלת רשימה של מספרים ומחזרים רשימה חדשה לאחר ביצוע left circular shift |

קוד

def deleting\_leftest\_number(number):

""" Function deleting\_leftest\_number gets a number

and return a string of that number without the leftest digit.

"""

return str(number)[1:]

def get\_leftest\_digit(number):

""" Function get\_leftest\_digit gets a number

and return the leftest digit of that number.

"""

return int(str(number)[0])

def concatenate\_unit\_digit\_to\_number(number, digit):

""" Function concatenate\_unit\_digit\_to\_number gets a number and a digit

and return a the number with the digit in its units.

"""

return int(str(number) + str(digit))

def left\_circular\_shift(list):

""" Function left\_circular\_shift gets a list of integers

and returns a left circular shift list.

"""

# declaring an empty list, that will contain the shifted list.

shifted\_list = []

# declaring a variable named "leftest\_of\_first\_element"

# that will contain the leftest digit of the element in index 0 in the list.

leftest\_of\_first\_element = 0

# running in a loop length of "list" times.

for i in range(len(list)):

# calculating and saving in a variable the leftest digit of the list[i].

leftest\_digit = get\_leftest\_digit(list[i])

# if its the first iteration.

if i == 0:

# saving the leftest digit of the first number for later on.

leftest\_of\_first\_element = leftest\_digit

else:

# changing the element in place of i - 1

# in the list to be

# shifted(leftest of list[i] to be digits of list[i-1]),

# by calling the function "concatenate\_unit\_digit\_to\_number".

shifted\_list[i - 1] = concatenate\_unit\_digit\_to\_number(

shifted\_list[i - 1], leftest\_digit)

# deleting the leftest digit in the number list[i]

# in order to complete the left circular shift of that cell in the list.

shifted\_element = deleting\_leftest\_number(list[i])

# appending to the list the complete shifted element.

shifted\_list.append(shifted\_element)

# dealing with the first cell that needs to add

# its leftest digit to the last cell to his units.

shifted\_list[-1] = concatenate\_unit\_digit\_to\_number(

shifted\_list[-1], leftest\_of\_first\_element)

# returning the complete shifted\_list.

return shifted\_list

# gets an input from the user of number of element.

number = int(input("Enter number of elements: "))

# declaring an empty list named "original\_array".

original\_array = []

# running in a loop "number" time.

for i in range(number):

# every iterating getting an input

# from the user about a element to add to the empty list.

original\_array.append(int(input("Enter element %s: " % i)))

# printing both lists, original and shifted.

print("\nOriginal array: \n%s" % original\_array)

print("\nShifted array: \n%s" % left\_circular\_shift(original\_array))

פלט לבעיה 2:

