שאלה 1

א) הפקודה lst = lst[::-1] אינה הופכת את סדר האיברים במקום. בשיטה הזאת פייתון למעשה יוצרת רשימה חדשה במקום חדש לגמרי בזיכרון, ומקשרת את השם lst אליה. זאת בניגוד להגדרה ל-in-place algorythem, שמשנה את הקלט שהוא קיבל.

ב) הפונקציה לא מחזירה שום פלט. היא מבצעת את הפעולות שלה במקום על הקלט שקיבלה.
הסיבה שאין פונקצית היפוך במקום למחרוזות, היא שמחרוזת היא עצם לא ניתן לשינוי (immutable), ולכן מהגדרתו אין פונקציה שתוכל לשנות מחרוזת במקום שבו היא נמצאת בזיכרון.

שאלה 2

א) ניתן לראות מהקוד שמספר האיטרציות תלוי אך ורק בערך של b, כלומר, הביטוי שלנו יהיה תלוי רק ב-m.
בקוד אנו כל פעם מחלקים את b בשתיים. זה שקול ל-לחלק את את הייצוג הבינארי של b ב-10. לפי חוקי החילוק, ברור שכשאנחנו מחלקים מספר ב10, אורך החלק השלם שלו יתקצר בספרה, כלומר בביט אחד.
לכן, מספר האיטרציות שהלולאה תרוץ עד שהערך השלם של המספר יהיה 0, הוא **m פעמים**.

בכל איטרציה יכולות להיות מקסימום שתי פעולות הכפלה, כפי שניתן לראות בקוד.
לכן, אנו כבר יכולים להסיק ש**מספר ההכפלות המקסימלי הוא 2m**, אם תמיד b%2 = = 1.
אם לעומת זאת תמיד b%2 =! 1, אז מספר ההכפלות יהיה שווה m.
אלה שתמיד המספר האחרון שנגיע אליו לפני ש b = 0 הוא b = 1, ואז יהיו שתי הכפלות.
לכן מספר ההכפלות המינימלי הוא m + 1.

שאלה 3

א) התשובה היא 3190 ביטים. ניתן לראות בתמונה, לפי הסדר, את שתי דרכי הפיתרון ((1) ו-(2)):

```
>>> math.floor(math.log2(eval("6**1234")) + 1)
3190
>>> len(bin(eval("6**1234"))[2:])
3190
```

ג) 1) המספר הגדול ביותר בעל חמש ספרות בבסיס עשר הוא 99999. המספר הקטן ביותר בבסיס זה הוא 10000. נציב בנוסחת הלוג ונקבל:
כלומר המספר המינימאלי הדרוש של ביטים לייצוג מספר כזה הוא **14**, והמכסימלי הוא **17.**

```
>>> math.floor(math.log2(99999)+1)
17
>>> math.floor(math.log2(10000)+1)
14
```

2) המספר הגדול ביותר בעל חמש ספרות בבסיס הקסדצימלי הוא fffff, והקטן ביותר הוא 10000.
נמיר את המספרים האלה למספרים בבסיס עשר, ונציב אותם בנוסחת הלוג:
כלומר מספר הביטים המינימאלי הוא **17**, ומספרם המכסימלי הוא **20.**

```
>>> fffff = 15*(16**4 + 16**3 + 16**2 + 16**1 + 16**0)
>>> hex10000 = 16**4 + 0 + 0 + 0 + 0
>>> math.floor(math.log2(fffff)+1)
20
>>> math.floor(math.log2(hex10000)+1)
17
```

ד) 1) לאחר שמוסיפים k אחדות לייצוג הבינארי של N, אז עבור כל ספרה "1" בייצוג זה, המעריך של הבסיס 2 גדל ב- k (לפי תוספת של k ביטים). לכן, נוכל להוציא גורם משותף 2^k מתוך כל אחד מהמחברים ה"יוצרים" את המספר N בייצוג הבינארי. סכ"ה N*(2^k).
לזה נצטרך להוסיף את k האחדות שנוספו. את זה נוכל לחשב פשוט ע"י נוסחת סכום סדרה הנדסית, ונקבל שהוא שווה ל- (2^k) – 1.
סכ"ה, הייצוג של המספר החדש יהיה **(2^k)*N +(2^k) – 1**

2) ע"פ נוסחאת הלוג, מספר הביטים בייצוג בינארי של מספר N הוא n = floor(logN + 1). כלומר, המעריך של הבסיס 2 האחרון במספר כזה יהיה n-1. לכן, כאשר אנו מוסיפים ספרת "1" נוספת מצד ימין, אנו למעשה מוסיפים למספר N את 2^n. ע"פ הנוסחא, נוכל לכתוב את הייצוג למספר החדש כך:
2^(floor(logN + 1)) + N

שאלה 4
א) ידוע לנו שיש אינסוף מספרים ראשוניים. לכן, קיימים אינסוף מספרים שמתחילים סדרות שמסתיימות באפס (כי מספרים ראשוניים מתחלקים רק בעצמם ובאחת).
קיומם של מספרים מושלמים מוכיחים שקיימות גם סדרות מסוג (2), למשל, 6 הוא התחלה של סדרה כזאת (סכום מחלקיו ממש הוא 6).

ב) עבור limit = 275 קיימים **268** מספרים שהם התחלה של סדרה סופית.

ג) כי אם נפספס ככה בטעות סדרה מסוג (3) לא נוכל לקבל מדליית פילדס.
אפשרות נוספת היא שסדרות מסוג (3) הן סדרות אינסופיות שאינן נכנסות למעגל מחזורי, כלומר הן בהכרח לא חסומות. אין לנו בשלב זה בקורס את הכלים המתמטיים להוכיח שסדרה כזאת קיימת, ומשום כך אם התוכנה שנבנה תיתקל בסדרה כזאת, היא תיאלץ לעבור איבר-איבר כדי לוודא שהסדרה אינסופית.
כלומר, התוכנה תיכנס ללופ שהיא לא תוכל לצאת ממנו.
אולי יעל לא רצתה שניתקע במצב כזה ולכן היא התנגדה להכנסת הדרישה הזאת. תודה יעל!