תרגיל 8 – מבני נתונים

תאריך פרסום: 28/12/2020

תאריך הגשה: 14/1/2021

מתרגל אחראי: אוריאל פרידמן

משקל התרגיל: 4 נקודות

הנחיות כלליות:

- העבודה תבוצע ביחידים.
- קראו את ההוראות לגבי הגשת תרגילי הבית באתר הקורס.
- מומלץ לקרוא את כל העבודה לפני תחילת הפתרון.
- כתבו תיעוד (הערות) שמסביר את הקוד שלכם. אין לכתוב הערות בעברית.
- עליכם להוריד את הקובץ "hw8.py" מתיקיית "תרגיל בית 8" מהמודל, ולהכניס את הקוד שלכם בשורות המתאימות. תבצעו את השינויים שלכם בהתאם להערות בקובץ.
- אין להשתמש בחבילות או במודולים, אלא אם נאמר במפורש.
- **ניתן להניח שהקלט תקין, אלא אם נכתב אחרת בשאלה.**
- העבודה תיבדק באופן אוטומטי ולכן על הפלטים להיות בדיוק כפי שמוגדר בתרגיל .
- העתקת קוד (מכל מקור שהוא) עלולה להוביל לכישלון בקורס. **אל תעתיקו!**
- שאלות בנוגע לעבודה ישאלו ב-"פורום שאלות לתרגיל בית 8" במודל או בשעות הקבלה של המתרגל האחראי בלבד.
- את העבודה יש להגיש דרך מערכת ההגשה בכתובת:
  https://subsys.ise.bgu.ac.il/submission/login.aspx
- טרם ההגשה אנא וודאו:
  - כל אחד מהקבצים מתקמפל ורץ כנדרש.
  - המשתנים שכתבתם עם שמות משמעותיים (ללא שמות כמו a).
  - בתחילת כל שאלה כתבתם הערות לקוד באנגלית.
  - אין הדפסות מיותרות (למשל הרצה של טסטים).
- אתם רשאים להוסיף שיטות לבחירתכם (בנוסף לשיטות שאתם נדרשים לממש בעבודה), בתנאי שהן אינן מפרות את הוראות הסעיף בו בחרתם לעשות בהן שימוש. על פונקציות העזר להיכתב בתוך המחלקות המתבקשות, אין לכתוב מחלקות נוספות.
- מטרות העבודה: תרגול מבני נתונים, בדגש על רשימות מקושרות ועצים, תוך תרגול כל החומר הנלמד במהלך הסמסטר.

בהצלחה!

מבוא - פולינום:

במתמטיקה פולינום ממשתנה X הוא ביטוי מהצורה: $a_0 + a_1x + a_2x^2 + . . . + a_nx^n$ כאשר $a_0, a_1, \ . . . \ , a_n$ הם קבועים.

המחוברים a_kx^k נקראים מונומים. במונום כזה k היא החזקה (או המעריך), והקבוע a_k הוא המקדם.

בעבודה זו נכתוב קוד המממש פעולות שונות על פולינומים.

חלק א' – מחלקת מונום:

המחלקה מונום (Monom) תהיה מחלקה המכילה את (ורק את) השדות הבאים:

- power – חזקת המונום (מספר טבעי)
- coef – מקדם (מספר ממשי)
- next – מצביע על חוליית המונום הבאה.

המחלקה תכיל את השיטות הבאות:

- בנאי המקבל כפרמטרים את החזקה והמקדם (ברירת מחדל של המקדם היא 1).
- פעולת __repr__ המחזירה מחרוזת המייצגת את המונום בפורמט הבא (לדוגמא עבור חזקה 6 ומקדם 3): 3X^6

  חשוב לשים לב לנקודות הבאות:

  - עבור החזקה 1 לא תודפס החזקה.
  - עבור חזקת 0 לא יודפס X (רק המקדם).
  - עבור מקדם 1 לא יודפס המקדם (לאור מורכבות המימוש עבור 1- כן).
  - אם במקדם הערך הלא שלם הוא 0, אז המקדם יודפס כמספר שלם.
  - אם במקדם הערך הלא שלם שונה מ 0 הוא יודפס עם **עד** 2 ספרות אחרי הנקודה - מעוגל, השתמשו בפונקציית round.
  - עבור מקדם 0 יודפס 0 בלבד.
  - עבור מקדם שלילי המונום ייוצג עם סוגריים.

לדוגמא עבור הקוד הבא:

```
m1 = Monom(6, 2)
m2 = Monom(8)
m3 = Monom(0, 19.5)
m4 = Monom(1, 5)
m5 = Monom(8, 1.0)
m6 = Monom(8, 0)
print('m1-m6:')
print('m1:', m1)
print('m2:', m2)
print('m3:', m3)
print('m4:', m4)
print('m5:', m5)
print('m6:', m6)
```

יודפס הפלט הבא:

```
m1-m6:
m1: 2X^6
m2: X^8
m3: 19.5
m4: 5X
m5: X^8
m6: 0
```

- __mul__ - כפל (בסקלר ובמונום אחר), הגדרת כפל מונום בסקלר היא החזרת מונום **חדש** שהמקדם שלו הוא המקדם של המונום שהוכפל כפול הסקלר, מכפלת מונומים תייצר ותחזיר מונום **חדש** שהמקדם שלו הוא מכפלת המקדמים והחזקה שלו הוא חיבור החזקות. בשני המקרים המכפלה אינה משנה את המונום עליה היא מופעלת, **יש לתמוך בכפל מימין.**

לדוגמא עבור הקוד הבא (בהמשך לקוד הקודם):

```
m7 = m1 * 3.5
m8 = 4 * m1
m9 = m7 * m8

print('m7-m9:')
print('m7:', m7)
print('m8:', m8)
print('m9:', m9)
```

יודפס הפלט הבא:

```
m7-m9:
m7: 7X^6
m8: 8X^6
m9: 56X^12
```

- נגזרת – שיטת derivative אינה מקבלת פרמטרים בקריאה, בהפעלת השיטה יוחזר אובייקט מונום חדש בערך הנגזרת של המונום עליו הופעלה הפעולה (בדומה לפעולת הכפל גם כאן לא ישתנה האובייקט עליו הופעלה הפעולה) נגזרת על מונום X מוגדרת כמונום חדש בו החזקה היא כשל X פחות 1 והמקדם הוא כשל המקדם של X כפול החזקה של X ($(ax^n)' = anx^{n-1}$).

לדוגמא עבור הקוד הבא (בהמשך לקוד הקודם):

```
m10 = m9.derivative()
m11 = m5.derivative()
m12 = m4.derivative()
m13 = m12.derivative()

print('m10-m13:')
print("m10=m9'(" + str(m9) + "'):")
print('m10:', m10)
print("m11=m5'(" + str(m5) + "'):")
print('m11:', m11)
print("m12=m4'(" + str(m4) + "'):")
print('m12:', m12)
print("m13=m12'(" + str(m12) + "'):")
print('m13:', m13)
```

יודפס הפלט הבא:

```
m10-m13:
m10=m9'(56X^12'):
m10: 672X^11
m11=m5'(X^8'):
m11: 8X^7
m12=m4'(5X'):
m12: 5
m13=m12'(5'):
m13: 0
```

- אינטגרל – שיטת integral אינה מקבלת פרמטרים בקריאה, בהפעלת השיטה יוחזר אובייקט מונום חדש בערך האינטגרל של המונום עליו הופעלה הפעולה (בדומה לפעולות הקודמות גם כאן לא ישתנה האובייקט עליו הופעלה הפעולה) אינטגרל על מונום X מוגדר כמונום חדש בו החזקה היא כשל X פלוס 1 והמקדם הוא כשל המקדם של X חלקי החזקה של X:

$$\int x^n \, dx = \frac{x^{n+1}}{n+1}$$

לדוגמא עבור הקוד הבא:

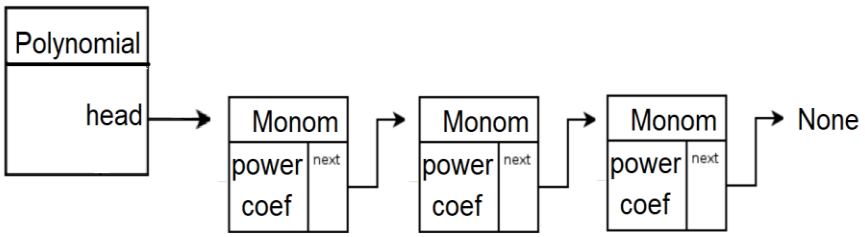
```
m14 = m10.integral()
m15 = m11.integral()
m16 = m12.integral()
m17 = m13.integral()
print('m14-m17:')
print('integral('+str(m10)+')='+str(m14))
print('integral('+str(m11)+')='+str(m15))
print('integral('+str(m12)+')='+str(m16))
print('integral('+str(m13)+')='+str(m17))
```

יודפס הפלט הבא:

```
m14-m17:
integral(672X^11)=56X^12
integral(8X^7)=X^8
integral(5)=5X
integral(0)=0
```

חלק ב' – מחלקת פולינום:

מחלקת פולינום - Polynomial מחלקה שתתאר פולינום כרשימה מקושרת של מונומים כמתואר באיור הבא:



איור זה הוא עבור מקרה של 3 חזקות שונות בפולינום, פולינום יכול להיות החל מפולינום ריק (head הוא None), ועד מספר גדול מאוד של מונומים.

מחלקת Polynomial תכיל את השדה הבא **בלבד**:

- head - ראש רשימה מקושרת של חוליות, הרשימה תהיה ממוינת לפי החזקות בסדר יורד (ראש הרשימה יהיה החזקה הגבוהה ביותר, לא יהיו שתי חזקות זהות בפולינום).

מחלקת פולינום תכיל את השיטות:

- בנאי __init__ - מקבל כפרמטר רשימה המכילה tuples של שני אברים מספריים כל אחד. בתוך ה tuple האיבר הראשון הוא החזקה והשני הוא המקדם של מונום מסוים.

  יש לטפל בקלט לא תקין:

  - הפרמטר אינו מסוג רשימה
  - לא כל האברים ברשימה מסוג tuple ובגודל 2
  - תכני ה - tuple אינם מספריים.

  בכל קלט לא תקין שהוא תועלה חריגה מסוג valueError עם הכיתוב (**המדויק**):

  'invalid polynomic initiation.'

  רשימה ריקה נחשבת תקינה, מונום עם מקדם 0 לא יתווסף לרשימה המקושרת של המונומים.

  המלצה – בנו שיטה המוסיפה מונום לפולינום ממוין.

- פעולת __repr__ - מחזירה מחרוזת המייצגת את הפולינום לפי הדוגמאות שבהמשך, פולינום ריק ייוצג כ '0'.

לדוגמא עבור הקוד הבא:

```
p1 = Polynomial([(1, 2), (2, 1), (5, 6), (6, 5), (1, 2), (2,
1)])
p2 = Polynomial([])
p3 = Polynomial([(8, 0), (3, -5), (3, 5), (0, 18)])
print('p1-p3:')
print(p1)
print(p2)
print(p3)
```

יודפס הפלט הבא:

```
p1-p3:
P(X)=5X^6+6X^5+2X^2+4X
P(X)=0
P(X)=18
```

- rank – דרגת הפולינום, ללא פרמטרים, תחזיר את החזקה הגבוהה ביותר של הפולינום שהמקדם שלה אינו 0 (עבור פולינום ללא מונומים החזקה תהיה 0).
- calculate_value – פעולה המקבלת ערך מספרי x כפרמטר ומחזירה את הערך המספרי המתקבל מהצבת x בפולינום.

לדוגמא עבור הקוד הבא (בהמשך לקוד הקודם):

```
print(p1, ' rank:', str(p1.rank()), )
print(p2, ' rank:', str(p2.rank()))
print(p3, ' rank:', str(p3.rank()))
print(p1, ' value(x=0):', str(p1.calculate_value(0)))
print(p1, ' value(x=1):', str(p1.calculate_value(1)))
print(p1, ' value(x=2):', str(p1.calculate_value(2)))
```

יודפס הפלט הבא:

```
P(X)=5X^6+6X^5+2X^2+4X  rank: 6
P(X)=0  rank: 0
P(X)=18  rank: 0
P(X)=5X^6+6X^5+2X^2+4X  value(x=0): 0
P(X)=5X^6+6X^5+2X^2+4X  value(x=1): 17
P(X)=5X^6+6X^5+2X^2+4X  value(x=2): 528
```

- פעולת __neg__ מחזירה פולינום חדש הזהה לפולינום עליו הופעלה הפעולה למעט סימן המקדמים שיהיה הפוך לפולינום עליו הופעלה הפעולה.
- פעולת __sub__ מחזירה פולינום חדש המכיל את פעולת חיסור הפולינום שהתקבל כפרמטר מהפולינום עליו הופעלה הפעולה.
- פעולת __add__ - המקבלת כפרמטר פולינום נוסף ומחזירה פולינום חדש שהוא תוצאת חיבור שני הפולינומים.

לדוגמא עבור הקוד הבא:

```
p4 = Polynomial([(1, 1), (2, 2), (3, 3)])
p5 = Polynomial([(1, 4), (2, 5), (3, 6)])
p6 = Polynomial([(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6,
6)])
p7 = Polynomial([(1, -1), (2, -2), (3, 3), (4, -4), (5, -5),
(6, -6)])
print('p4-p7(v1)')
print('p4:', p4)
print('p5:', p5)
print('p6:', p6)
print('p7:', p7)
print(p6, '+', p7, '=', (p6+p7))
print(p4, '+', p5, '=', (p4+p5))
print('-(', p6, ')=', -p6)
print(p6, '-', p7, '=', (p6-p7))
```

יודפס הפלט הבא:

```
p4-p7(v1)
p4: P(X)=3X^3+2X^2+X
p5: P(X)=6X^3+5X^2+4X
p6: P(X)=6X^6+5X^5+4X^4+3X^3+2X^2+X
p7: P(X)=(-6X^6)+(-5X^5)+(-4X^4)+3X^3+(-2X^2)+(-1X)
P(X)=6X^6+5X^5+4X^4+3X^3+2X^2+X + P(X)=(-6X^6)+(-5X^5)+(-4X^4)+3X^3+(-2X^2)+(-1X) = P(X)=6X^3
P(X)=3X^3+2X^2+X + P(X)=6X^3+5X^2+4X = P(X)=9X^3+7X^2+5X
-( P(X)=6X^6+5X^5+4X^4+3X^3+2X^2+X )= P(X)=(-6X^6)+(-5X^5)+(-4X^4)+(-3X^3)+(-2X^2)+(-1X)
P(X)=6X^6+5X^5+4X^4+3X^3+2X^2+X - P(X)=(-6X^6)+(-5X^5)+(-4X^4)+3X^3+(-2X^2)+(-1X) = P(X)=12X^6+10X^5+8X^4+4X^2+2X
```

- פעולת כפל __mul__ - המקבלת סקלר או פולינום, פעולת הכפל מוגדרת כך:
  - כפל פולינום בסקלר - יצירת פולינום חדש שכל המקדמים בו הם המקדמים של הפולינום המקורי כפול הסקלר.
  - כפל פולינום בפולינום – יצירת פולינום חדש שהוא חיבור מכפלת כל מונום מאחד הפולינומים בפולינום השני (פעולה קומוטטיבית), עוד על כפל פולינומים ניתן למצוא כאן.
  - **יש לתמוך בכפל מצד ימין.**
  - המלצה – כדאי לממש כפל פולינום במונום.

לדוגמא עבור הקוד הבא (בהמשך לקוד הקודם):

```
print('p4-p5(v2)')
print('p4:', p4)
print('p5:', p5)
print(p4, '*', p5, '=', (p4*p5))
print(p4, '*', 5, '=', (p4*5))
print(5, '*', p4, '=', (5*p4))
```

יודפס הפלט הבא:

```
p4-p5(v2)
p4: P(X)=3X^3+2X^2+X
p5: P(X)=6X^3+5X^2+4X
P(X)=3X^3+2X^2+X * P(X)=6X^3+5X^2+4X = P(X)=18X^6+27X^5+28X^4+13X^3+4X^2
P(X)=3X^3+2X^2+X * 5 = P(X)=15X^3+10X^2+5X
5 * P(X)=3X^3+2X^2+X = P(X)=15X^3+10X^2+5X
```

- נגזרת - derivative - השיטה תחזיר פולינום חדש המכיל רשימה שכל אבר בה הוא הנגזרת של מונום של הרשימה בפולינום המקורי.
- אינטגרל – integral – השיטה תחזיר פולינום חדש המכיל רשימה שכל אבר בה הוא האינטגרל של מונום של הרשימה בפולינום המקורי. השיטה תקבל משתנה מספרי (ברירת מחדל 0) שיהיה הקבוע (X בחזקת 0) בפולינום החדש.

לדוגמא עבור הקוד הבא (בהמשך לקוד הקודם):

```
print('p4-p5(v3)')
print('(', p4, ")'=", p4.derivative())
print('(', p5, ")'=", p5.derivative())
print('integral(', p4, ")=", p4.integral())
print('integral(', p5, ")=", p5.integral())
print('integral(', p5, ")-18=", p5.integral(-18))
print('integral((', p4, ")')=", p4.derivative().integral())
```

יודפס הפלט הבא:

```
p4-p5(v3)
( P(X)=3X^3+2X^2+X )'= P(X)=9X^2+4X+1
( P(X)=6X^3+5X^2+4X )'= P(X)=18X^2+10X+4
integral( P(X)=3X^3+2X^2+X )= P(X)=0.75X^4+0.67X^3+0.5X^2
integral( P(X)=6X^3+5X^2+4X )= P(X)=1.5X^4+1.67X^3+2X^2
integral( P(X)=6X^3+5X^2+4X )-18= P(X)=1.5X^4+1.67X^3+2X^2+(-18)
integral(( P(X)=3X^3+2X^2+X )')= P(X)=3X^3+2X^2+X
```

- השוואות – יש לתמוך בכל ששת פעולות ההשוואה (שימו לב שאין צורך לממש את כולן), פולינום יחשב ליותר גדול מפולינום אחר אם בדרגה הגבוהה ביותר השונה בין שני הפולינומים יש לו מקדם גבוה יותר.

(פולינומים שווים הם פולינומים המכילים בדיוק את אותם ערכים בכל המונומים).

לדוגמא עבור הקוד הבא:

```
print('p8-p9')
p8 = Polynomial([(1, 4), (2, 5), (3, 6)])
p9 = Polynomial([(1, 4), (2, 5), (3, 6)])
p10 = Polynomial([(1, 4.1), (2, 5), (3, 6)])
p11 = Polynomial([(4, 5)])
print(p8, '==', p9, ':', str(p8 == p9))
print(p8, '<=', p9, ':', str(p8 <= p9))
print(p8, '>=', p9, ':', str(p8 >= p9))
print(p8, '<', p9, ':', str(p8 < p9))
print(p8, '>', p9, ':', str(p8 > p9))
print(p8, '!=', p9, ':', str(p8 != p9))
print('p9-p10')
print(p9, '==', p10, ':', str(p9 == p10))
print(p9, '<=', p10, ':', str(p9 <= p10))
print(p9, '>=', p10, ':', str(p9 >= p10))
print(p9, '<', p10, ':', str(p9 < p10))
print(p9, '>', p10, ':', str(p9 > p10))
print(p9, '!=', p10, ':', str(p9 != p10))
print('p9-p11')
print(p9, '==', p11, ':', str(p9 == p11))
print(p9, '<=', p11, ':', str(p9 <= p11))
print(p9, '>=', p11, ':', str(p9 >= p11))
print(p9, '<', p11, ':', str(p9 < p11))
print(p9, '>', p11, ':', str(p9 > p11))
print(p9, '!=', p11, ':', str(p9 != p11))
```

יודפס הפלט הבא:

```
p8-p9
P(X)=6X^3+5X^2+4X == P(X)=6X^3+5X^2+4X : True
P(X)=6X^3+5X^2+4X <= P(X)=6X^3+5X^2+4X : True
P(X)=6X^3+5X^2+4X >= P(X)=6X^3+5X^2+4X : True
P(X)=6X^3+5X^2+4X < P(X)=6X^3+5X^2+4X : False
P(X)=6X^3+5X^2+4X > P(X)=6X^3+5X^2+4X : False
P(X)=6X^3+5X^2+4X != P(X)=6X^3+5X^2+4X : False
p9-p10
P(X)=6X^3+5X^2+4X == P(X)=6X^3+5X^2+4.1X : False
P(X)=6X^3+5X^2+4X <= P(X)=6X^3+5X^2+4.1X : True
P(X)=6X^3+5X^2+4X >= P(X)=6X^3+5X^2+4.1X : False
P(X)=6X^3+5X^2+4X < P(X)=6X^3+5X^2+4.1X : True
P(X)=6X^3+5X^2+4X > P(X)=6X^3+5X^2+4.1X : False
P(X)=6X^3+5X^2+4X != P(X)=6X^3+5X^2+4.1X : True
p9-p11
P(X)=6X^3+5X^2+4X == P(X)=5X^4 : False
P(X)=6X^3+5X^2+4X <= P(X)=5X^4 : True
P(X)=6X^3+5X^2+4X >= P(X)=5X^4 : False
P(X)=6X^3+5X^2+4X < P(X)=5X^4 : True
P(X)=6X^3+5X^2+4X > P(X)=5X^4 : False
P(X)=6X^3+5X^2+4X != P(X)=5X^4 : True
```

<u>דגשים למחלקת פולינום:</u>

- בכל השיטות (למעט הבנאי או שיטות נוספות שלכם) אין לשנות את הפולינום (או פולינומים) המשתתפים בשיטה.
- בכל הפעולות, אם נוצר מונום שערך המקדם בו 0 – אין להכליל את המונום ברשימת המונומים של הפולינום.
- בכל הפעולות – כולל הבנאי, אם נוצרו שתי חוליות בעלי חזקה זהה יש לחבר אותן למונום יחיד שהמקדם שלו הוא סכום המקדמים.
- ניתן להוסיף שיטות כרצונכם כל עוד הם משתמשות בחומר שלמדנו בלבד, חובה לשמור על הפונקציונאליות של השיטות שפורטו – כך ששיטות נוספות יהיו שיטות עזר בלבד.

חלק ג' – עצי מיון בינאריים של פולינומים:

נתונה בקוד מחלקת חוליית עץ בינארי BinTreeNode:

```
class BinTreeNode:
    def __init__(self, val):
        self.value = val
        self.left = self.right = None
```

יש לממש מחלקת עץ מיון בינארי לפולינומים: PolynomialBST – שבדומה לעץ חיפוש תכיל:

שדה:

- head – ראש העץ, מכיל None בעץ ריק או מצביע לחוליה הראשונה שצורפה לעץ, החוליות יהיו ללא key, ההשוואה תהיה לפי פעולות ההשוואה השונות לכל פולינום.

שיטות:

- שיטת insert – הכנסת חולייה המכילה פולינום לעץ בצורה ממוינת, כך שהכנסת חולייה המכילה פולינום קטן או שווה לחולייה המכילה פולינום קיים יהיה ברגלה השמאלית של חוליית הפולינום הקיים.
- שיטת in_order – החזרת העץ (פולינומים בלבד) לפי tree traversal – in order, ברשימה, עבור עץ ללא חוליות יש להחזיר רשימה ריקה.
- שיטת __add__ – המקבלת כפרמטר עץ בינארי אחר other, יש להחזיר עץ בינארי חדש המכיל את אברי שני העצים, אין לשנות את העצים המשתתפים בחיבור.

לדוגמא עבור הקוד הבא:

```
print('t1')
t1 = PolynomialBST()
t1.insert(p1)
t1.insert(p2)
t1.insert(p3)
t1.insert(p4)
t1.insert(p5)
print(t1.in_order())
print('t2')
t2 = PolynomialBST()
print(t2.in_order())
t2.insert(p6)
t2.insert(p7)
t2.insert(p8)
t2.insert(p9)
t2.insert(p10)
t2.insert(p11)
print(t2.in_order())
print('t3')
t3 = t1 + t2
print(t3.in_order())
```

יתקבל הפלט הבא:

```
t1
[P(X)=0, P(X)=10, P(X)=3X^3+2X^2+X, P(X)=6X^3+5X^2+4X, P(X)=5X^6+6X^5+2X^2+4X]
t2
[]
[P(X)=6X^3+5X^2+4X, P(X)=6X^3+5X^2+4X, P(X)=6X^3+5X^2+4.1X, P(X)=5X^4, P(X)=(-6X^6)+(-5X^5)+(-4X^4)+3X^3+(-2X^2)+(-1X), P(X)=6X^6+5X^5+4X^4+3X^3+2X^2+X]
t3
[P(X)=0, P(X)=10, P(X)=3X^3+2X^2+X, P(X)=6X^3+5X^2+4X, P(X)=6X^3+5X^2+4X, P(X)=6X^3+5X^2+6X, P(X)=6X^3+5X^2+4.1X, P(X)=5X^4, P(X)=(-6X^6)+(-5X^5)+(-4X^4)+3X^3+(-2X^2)+(-1X), P(X)=5X^6+6X^5+2X^2+4X, P(X)=6X^6+5X^5+4X^4+3X^3+2X^2+X]
```

בהצלחה !!!! אוריאל.