מטלה – לולאות למתקדמים

במטלה זו נלמד לבנות מבנים "דמויי-מיכלים", שאפשר לרוץ עליהם בלולאה למרות שאינם תופסים מקום בזיכרון. השם הרשמי של מבנה מסוג זה הוא `iterable`. נבנה חמישה דמויי-מיכלים (מהקל אל הקשה):

א. **range** – מייצג טווח של עצמים. לדוגמה, `range(a,b)`, כאשר a ו-b הם מספרים שלמים, מייצג את המספרים השלמים מ-a עד b, כולל a ולא כולל b. לדוגמה, `range(5,9)` מייצג את המספרים 5,6,7,8. ניתן להניח ש b>a.

ב. **chain –** מייצג שירשור של שני דמויי-מיכלים. לדוגמה, `chain(range(1,4),range(5,9))` מייצג את המספרים 1,2,3,5,6,7,8. שימו לב: כל מיכל הוא בפרט גם דמוי-מיכל. בפרט, גם `string` (מחרוזת) הוא דמוי-מיכל, ולכן `chain` צריך לעבוד גם עליו. לדוגמה, הביטוי:

```
chain(range('a','d'),string("hello"))
```

מייצג את סדרת האותיות `a,b,c,h,e,l,l,o`.

ג. **zip –** מייצג חיבור במקביל של שני דמויי-מיכלים. ניתן להניח שהם באותו אורך. לדוגמה, הביטוי:

```
zip(range(1,6),string("hello"))
```

מייצג סדרה של חמישה זוגות סדורים:

```
1,h   2,e   3,l   4,l   5,o
```

ד. **product** – מייצג מכפלה קרטזית של שני דמויי-מיכלים (לא דווקא באותו אורך). לדוגמה:

```
product(range(1,4),string("helo"))
```

מייצג 12 זוגות סדורים:

```
1,h  1,e  1,l 1,o  2,h  2,e  2,l  2,o  3,h  3,e  3,l  3,o
```

ה. **powerset –** מייצג את כל תת-הקבוצות של דמוי-מיכל. לדוגמה:
`powerset(string("abc"))` מייצג 8 תת-קבוצות:

```
{},{a},{b},{a,b},{c},{a,c},{b,c},{a,b,c}
```

דגשים:

- יש לפתור כל סעיף ע"י פונקציה אחת המטפלת בכל סוגי-הנתונים – מספרים, תוים, מספרים-פיסיקליים, וכו' (למשל בסעיף א, יש לכתוב רק פונקציה range אחת – אין לכתוב פונקציות שונות למספרים ולתוים).
- יש לפתור כל סעיף בקובץ hpp נפרד; שמות הקבצים כתובים בקובץ Demo.cpp.
- יש חשיבות לסדר האיברים בתוצאה. ראו דוגמאות בקובץ Demo.cpp.
- יש להקפיד על כללי הנדסת תוכנה, ובפרט: קוד קריא, תיעוד ובדיקות-יחידה מקיפות.

- יש לחזור על החומר של ההרצאות לפני שמתחילים לכתוב, ולהשתמש בו לפי הצורך.
- מותר להשתמש בתכונות מתקדמות של שפת C++ גם אם עדיין לא נלמדו בהרצאות.
- אין להעתיק תרגילים שלמים מסטודנטים אחרים. מותר להיעזר בקטעי קוד מהאינטרנט, אולם **יש לציין בבירור את המקור**, לוודא שהקוד עובד, ולוודא שאתם מבינים למה הוא עובד.

שלבי העבודה

בשלב א, עליכם לכתוב קובץ Test.cpp הכולל בדיקות-יחידה מקיפות, באותו אופן של קבצי הבדיקות מהמטלות הקודמות (בעזרת הbadkan.hpp). כמו כן יש לכתוב את הקבצים הדרושים על-מנת שהפקודות הבאות ירוצו בלי שגיאות קימפול (לא חייבים לעבור את כל הבדיקות):

```
make demo && ./demo
make test && ./test
```

בשלב ב, יש לכתוב מימוש מלא העובר את כל הבדיקות – שלכם ושלנו.

הגשה לבדיקה אוטומטית

צרו מאגר (repository) חדש בגיטהאב והעלו לשם את הקבצים בתיקיה הראשית.

הגישו בטופס-ההגשה קישור-שיבוט למאגר - הקישור שרואים כשלוחצים על הכפתור clone בגיטהאב.