תרגיל מס׳ 3 – ״מחשבון קבוצות – וטיפול בחריגות״

**שמות מגישים:**

נדב אטיס - 316092667

אריאל אמון- 208440308

תיאור התרגיל:

בתרגיל זה מימשנו "מחשבון קבוצות". המחשבון מחשב את תוצאות הפעולות המבוקשות על קבוצות (מתמטיות, ללא כפילויות) של מספרים שלמים לפי הקלט מהמשתמש – קוד זה בניגוד לתרגיל ה-1 מתבסס על הפתרון של בית הספר. ומתוך כך על מבני הנתונים ועל התיכון אין אנו יכולים לתת את הדעת בצורה מלאה לגמרי – אם כי כפי שיתברר מטה, הבנו כנדרש.

**במקרה של טיפול בחריגות אנא שימו לב לגישות שבחרנו**

\***במקרה והשורה מכילה מספר גדול של ארגומנטים אך תחילית הפקודה תקינה נקבל את הפקודה ונתעלם מהשאר** \* (לדוגמה : uni 0 0 abc asd – נחשב uni 0 0 אך מהשאר נתעלם.)

\***בכל מקרה של שגיאה בקובץ אנו מנהלים שיח דרך הטרמינל עם המשתמש (גם ב-resize) \***

**\* שימו לב כאשר מזינים את הפקודה read ו resize על הקלט להיות רציף כלומר : resize 6\* או read ונתיב הקובץ**

**\* גם במקרה של קלט מהקובץ וגם במקרה של קלט מה- shell**

**בחרנו לקרוא את כל השורה ב״וואן שוט״ ורק אז לטפל בשגיאות אם קיימות.**

**במקרה בו נכנסת מספר פקודה לא תקינה כדוגמת uni -1 3 נחכה לסיום הקלט ורק אז נדפיס הודעת שגיאה מתאימה.**

המשתמש יכול גם ליצור פעולות מורכבות יותר ולהוסיף אותן למחשבון ואחר כך לבקש חישוב של תוצאה של הפעלתן על קבוצות מסוימות.

ממשק התרגיל מתבצע דרך הטרמינל.

**תיכון :(design)**

**מחלקתSetCalculator:**

זוהי המחלקה הראשית שמנהלת את ממשק המחשבון – מחזיקה בתוכה וקטור של מצביעים חכמים (shared\_ptr) שמחזיק את הפעולות האפשריות כעת בתפריט.

מחלקה זו אחראית לתפיסת כל החריגות שקורות במהלך ריצת התוכנית ותופסת סוגים רבים של חריגות שמקורם בספרייה הסטנדרטית. למשל חריגה ביחס לשם שגוי של קובץ (חריגה הנשלחת ע״י הדלקת הדגלים failbit או badbit ).

סוגי החריגות שנשלחות בכל אחד מהמקרים –

**Invalid\_argument** – נשלחת ביחס ל- מחיקה של אובייקט כשהגענו למינימום (3)

אם נכנסו מעט מדי ארגומנטים.

המשתמש ניסה להוסיף פעולה כשהמכסה שהוא קבע אוסרת עליו.

מספר הפעולה שגוי (eval -1).

פעולה שלא קיימת כלל (div).

**out\_of\_range -**  אם ניסנו להגביל את מספר הפעולות מחוץ לתחום הקבוע (100 – 3 )

כל שאר החריגות שנשלחות נשלחות ע״י stdl ונתפסות ע״י failure.

אחראית לניווט בין במחלקות השונות ומשתמש במטודת הפולימורפיזם כדי לעשות זאת.

**מחלקת Operation:**

מחלקת בסיס אבסטרקטית – אם לפעולות השונות. אובייקטים רבים יורשים ממנה. איננה מכילה ממברים פרטיים. יש בה פונקציות מעטפת החיוניות לכל מחלקה שיורשת ממנה(כדוגמת הדפסה וכו׳).

**מחלקות Union , Comp , Product ,Difference , ו Intersection:**

מחלקות שיורשות מ- Operation ומכילות בתוכם מימוש של הפעולה החופפת לשמם.

**מחלקת BinaryOperation:**

מחלקה שיורשת מ- Operation. כשמה כן היא – מהווה בסיס למחלקת הפעולות המורכבות – אלה שאחראים לפעולות על פעולות בסיסיות.

מחזיקה שתי מצביעים חכמים מסוג shared\_ptr שמחזיקות בתוכם מצביע פעולה המוגדרת מימין ומשמאל לפועלת ה-complex.

כוללות מימוש עוקף ל- eval

**מחלקת NameGenerator:**

מכולל אותיות גנרי.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

רשימת הקבצים:

**main.cpp**

**SetCalculator.cpp** - ניהול שוטף + טיפול בחריגות.

**Operation.cpp** - מחלקת בסיס אבסטרקטית.

**Difference.cpp - אחראית לביצוע פעולת החיסור בין קבוצות.**

**Intersection.cpp** -  **אחראית לביצוע פעולת החיתוך בין קבוצות.**

**Union.cpp**  - **אחראית לביצוע פעולת האיחוד בין קבוצות.**

**NonCompBinaryOperation.cpp**  - **אחראית לביצוע פעולת הרכבה בין קבוצות.**

**Product.cpp** - אחראית לביצוע כפל בין **פעולות**.

**Comp.cpp**  - אחראית לביצוע הרכבה בין **פעולות**.

**NameGenerator.cpp**  - **מחולל שמות קבוצה גנרי.**

**Identity.cpp**

לכל קובץ cpp קיים קובץ h ולכן סך הכל קיימים 11 קבצי cpp 10 קבצי h.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

מבנה נתונים:

**אותם מבני נתונים שיחזקאל עשה איתם שימוש ללא שום שינויץ**

אלגוריתם לציין:

**הקוד לא שונה כלל מהקוד המקורי של יחזקאל. הוספנו רק ניהול חריגות ואת כולם ביצענו מרמה גבוהה.**

**באגים ידועים:**

**לא קיימים כאלה.**