Assignment 1 – Part 1

Yuval Pariente, Maia Barber

שאלה 1.1

1.

א. תכנות אימפרטיבי

התכנית היא רצף מפורש של פקודות והיא קורית ביצוע של הפקודות זו אחר זו.

ב. תכנות פרוצדורלי

מגדירים קוד כפרוצדורה כך שניתן לקרוא ממקומות שונים בקוד.

ג. תכנות פונקציונלי

שימוש בסדרת ביטויים כך שהרצת התכנית הינה חישוב ומציאת ערך של הביטויים, ולא ביצוע של פקודות.

2. יתרונות התכנות הפרוצדורלי על התכנות האימפרטיבי
    1. תכנות פרוצדורלי מאפשר שימוש חוזר בפונקציות, מה שגורם לשיפור בקריאות ותחזוקת הקוד.
    2. חלוקת הקוד לפונקציות מאפשרת קוד מאורגן יותר מאשר רצף פקודות זו אחר זו.
    3. קל לבדוק פונקציה מסויימת מאשר את כל הלוגיקה של הקוד כולו.
3. יתרונות התכנות הפונקציונלי על התכנות הפרוצדורלי
    1. קל יותר לבצע תהליכים באופן מקבילי.
    2. אין side effect.
    3. הקוד מופשט ומעוצב יותר.
    4. קל יותר לבצע בדיקות ואימות.

שאלה 1.2

```
function calculateRevenueByCategoryFP(transactions: Transaction[]): Record<string, number>{

  return transactions.map(transaction => { const total = transaction.quantity > 5 ?
  transaction.price * transaction.quantity * 0.9  : transaction.price * transaction.quantity;

      return { category: transaction.category, price: transaction.price, quantity: transaction.quantity, total: total };

}).filter(transaction => transaction.total >= 50).reduce((acc, transaction) => {

    if (!acc[transaction.category])

                acc[transaction.category] = 0;

    acc[transaction.category] += transaction.total;

    return acc;

}, {} as Record<string, number>);
```

הערות:

1. Map - לכל עסקה נבדוק אם נמכרו יותר מ5 יחידות: אם כן, תינתן הנחה של 10 אחוז, אם לא, לא תינתן הנחה. נחשב את הסכום הכולל של כל עסקה לפי כמות * מחיר, ואם יש הנחה או לא.
2. Filter – אחרי שחישבנו את הסכום לכל עסקה (בהתחשב בהנחה או לא), נסנן החוצה את העסקאות בהן הסכום הכולל היה קטן מ 50.
3. Reduce – ניצור אובייקט שמכיל את הסכום הכולל לכל קטגוריה, לכל עסקה נעבור ונסכום בהתאם לקטגוריה אליה היא שייכת.

שאלה 1.3

1. <T>(x: T[], y: (item: T) => Boolean) => Boolean

// הפונקציה בודקת כל איבר במערך ומחזירה TRUE אם לפחות איבר אחד עובר את התנאי של הפונקציה הבוליאנית Y.

2. (x: number[]) => number[]

// פעולת הכפל המוגדרת מגבילה לנו את הטיפוס של Y למספרים בלבד, הפונקציה מחזירה מערך חדש של מספרים.

3. <T>(x: T[], y: (item: T) => Boolean) => T[]

// הפונקציה מקבלת מערך ופונקציית בדיקה בוליאנית ומחזירה מערך של כל האיברים שקיבלו TRUE בפונקציית הבדיקה.

4. (x: number[]) => number

// הפונקציה סוכמת בין acc, cur ולבסוף מחזירה מספר.

5. <T>(x: boolean, y: [T,T]) => T

// X הוא תנאי בוליאני, Y הוא מערך עם שני איברים ששניהם מסוג T (כיוון ומערכים חייבים להיות הומוגניים. אם התנאי הוא TRUE הפונקציה תחזיר Y[0], אם FALSE הפונקציה תחזיר Y[1]. לכן לבסוף הפונקציה תחזיר T.

6. <T1,T2>(f: (arg: T1) => T2, g: (n: number) => T1) => (x: number)

// X הוא מספר בגלל שעושים איתו פעולת חיבור, g היא פונקציה שמקבלת מספר ומחזירה T1, F מקבלת את הפלט של g כלומר T1, ומחזירה T2.

*ההערות נועדו בשבילנו, שנקבל רפרנס למה שעשינו כשנחזור על המטלות לקראת סוף הסמסטר.