חלק 1

שאלה 1:

סעיף a:

התבנית האימפרטיבית מתייחסת לתוכנית כרצף של פקודות מפורשות, והרצת התוכנית היא למעשה ביצוע של הפקודות בזו אחר זו.

סעיף b:

התבנית הפרוצדורלית מוסיפה לתבנית האימפרטיבית יכולות מגוונות:

- אריזה של קטע קוד שחוזר על עצמו בפרוצדורה מוגדרת עם קלט ופלט (פונקציות).
- שימוש במשתנים בתוך סקופ מקומי.
- לולאות.

השיפור על גבי התבנית האימפרטיבית מתבטא בכך שהוספנו הפשטה ומבניות שמקלה על כתיבת הקוד וקריאתו.

סעיף c:

בתבנית הפונקציונאלית, התוכנית היא סדרת ביטויים, והרצתה זה חישוב של הביטויים ומציאת ערכם.

- מאפשרת הרכבה של פונקציות
- לא מעדכנים מצבים משותפים - אין תופעות לוואי

זהו שיפור על גבי התבנית הפרוצדורלית במובן שקל יותר לאמת קוד, ולמקבל עם מספר ת'רדים.

שאלה 2:

```
function averageGradesOver60(grades: number[]) {
    const filteredGrades = grades.filter((x: number) => x > 60);
    return filteredGrades.reduce((acc: number,curr: number) => acc + curr/filteredGrades.length, 0);
}
```

שאלה 3:

סעיף a:

$$< T > (x: T[\,], y: (z: T) => boolean) => boolean$$

סעיף b:

$$(x: number[\,]) => number$$

סעיף c:

$$< T > (x: boolean, y: T[\,]) => T$$

סעיף d:

פעולת + מוגבלת למספרים בלבד בעבודה, לכן x הוא מסוג number.

g מקבלת מספר ומחזירה ערך כלשהו ש-f מקבלת כפרמטר.

נסמן את החתימה של g: $(y: number) => T$

ואת החתימה של f: $(z: T) => U$

אז לסיכום:

$$< T, U > (\, f: (z: T) => U, g: (y: number) => T\,) => number => U$$

שאלה 4:

Abstraction barriers – הפרדה של "שכבות" שונות של המערכת, על ידי בין היתר *encapsulation* של פונקציות. בשכבה החיצונית המערכת קוראת לפונקציות משכבות פנימיות מבלי לדעת מה אופן מימושן.