PPL- עבודה 1

1)

<u>פרדיגמה אימפרטיבית</u>- תוכניות הבנויה על רצף של פקודות, כאשר הפקודות מתבצעות באופן מסודר אחת אחרי השנייה. כמו כן, מאופיינת במשתנים, פקודות ומבני בקרה.

<u>פרדיגמה פרוצדורלית-</u> תוכנית שמחולקת לתת תוכניות רבות אשר כל תת תוכנית נקראת פרוצדורה בפני עצמה . לכל פרוצדורה יש משימה שלשמה היא מוגדרת. אוסף של פקודות שעובדות בסדר מסוים, לעיתים יחזירו ערך ולעיתים לא, שהינן חלק ממכלול גדול המרכיב את התוכנית. שפה פרוצדורלית מוכרת- C.

<u>פרדיגמה פונקציונלית-</u> פרדיגמה שמתנהגת כמו פונק׳ מתמטית. התוכנית היא ביטוי או סדרת ביטויים ולא רצף של פקודות. הרצת התוכנית היא חישוב של הביטויים. כלומר, מציאת הערך ולא ביצוע של ציווים. כמו בפונקציה מתמטית, אם נקבל ערך , קיים לו ערך/ פלט שמתאים לו יחיד.

הפרדיגמה הפרוצדורלית משפרת את האימפרטיבית בכך שהתוכנית הפרוצדורלית מונעת חזרות של קוד שלא לצורך, למשל לולאת for וכו׳ שמונעת מאיתנו לכתוב את אותו הדבר מס׳ פעמים. תחביר השפה מלמד שמדובר בדפוס חוזר, כמו כן, מאפשר קוד היררכי של הקוד האימפרטיבי.

הפרדיגמה הפונקציונלית משפרת את הפרוצדורלית בכך שקיימת העברת נתונים מפונק׳ לפונק׳ ועל ידי כך ניתן לשפר את המקביליות. בנוסף הפרדיגמה הפונקציונלית משפרת את האופטימיזציה של התוכנית.

2)

א. x הינו מסוג מערך מכל סוג . y הינה פונק׳ callback שמחזירה ערך בוליאני
ב. x – מערך של מספרים, כאשר הצובר המתחיל מ-0 מרמז לנו על מספרים, והצובר והנוכחי מצביעים על הצבירה, כאשר curr הוא איבר נוכחי במערך שמתווסף לצובר.
ג. x – משתנה /תנאי בוליאני y הוא מערך.

3) abstraction barriers- הרעיון הוא להשתמש בפונק׳ שנכתבו ע״י מתכנתים אחרים, ללא ידיעה בהכרח איך מומשה הפונקציה או כיצד היא פועלת. המתכנת סומך על נכונות המימוש ובונה תוכנית ופונקציות חדשות על סמך הקיימות. ומשתמש בעיקר על סמך ידיעה מה הקלט, מה דרך הפעולה שלהן ומה ערך ההחזרה אם קיים כזה. כלומר, באופן כללי הרעיון הוא הפרדת השכבות של התכנות.