# תרגיל בית 5 בשפות תכנות


# שאלה 1

א.מנגנון הRTTI או בשמו המלא Run time type information הוא מנגנון שמכיל מידע על הטיפוס במן ריצה. למשל הוא יכול להכיל את שם הטיפוס, דוגלו בזיכרון, הפונקציות והשדות שלו וכו'...
ב. בשפת c לא קיים מנגנון RTTI. בגלל גישת ה "no hidden cost" שיש לשפה.
ג.מנגנון איסוף האשפה צריך לדעת בזמן ריצה מה גודלו של האוביקטים אותו הוא צריך לשחרר על מנת לדעת לדעת כמה בתים בזכרון עליו לשחרר. מידע זה ימצא בRTTI והמנגנון איסוף אשפה יכול להשתמש בו לצורך זה.
ד.שימוש אחר לRTTI בשפות הדוגלות בstatic typing יכול להיות גם עבור deep clooningשל טיפוסים שיאפשר לדעת את הגודל/רכיבים של הטיפוס.

# שאלה 2

א. שונה-
static array מוקצה על הdata segmant לעומת dynamic array שמוקצה על הheap.
dynamic array מוקצה בזמן ריצה ואילו השני מוקצה בזמן קומפילציה
גודל המערך לא ידוע בזמן קומפילציה עבור dynamic array ועבור static array ידוע.
קבוצת האינדקסים נקבעת בזמן יצירה ב dynamic array ובזמן יצירה static array.
דומה-
שניהם מערכים התומכים בפעולות המוכרות
לאחר הקצאה אין אפשר לשנות את גודלם
ב.שונה-
גודלו של הstack based array נקבע בזמן ריצה ולא יכול להשתנות לאחר מכן, אך גודלו של associative array משתנים כשערכים מוספים או מוסרים ממנו.
associative array קבוצת הערכים יכולה להיות מכל טיפוס לעומת קבוצת הערכים של stack based array.
דומה-
שניהם מערכים
גודלם של המערכים נקבעים הזמן ריצה
ג. מנגנון הG1:

1) מחלק את הheap להרבה חלקים קטנים (יותר מ2 כמו שעושים חלק מהמנגנונים האחרים)
2) כאשר יש חלקים מלאים או כאלו שעומדים להתמלא, המנגנון עובר על החלקים הללו ומפנה מהם את האשפה
3) אם הוא פינה את החלק כולו הוא מסמן אותו כחלק ריק.

היתרון של מנגנון זה הוא שהוא לא עובר על כל הheap בכל איטרציה כמו אלו שראינו בכיתה על לבדוק אילו משתנים לא משתמשים בהם כבר, אלא הוא עובר רק על חלקים קטנים מheap כי שם יש יותר סיכוי שיצטרך לעשות עבודה.
לעומת זאת, חסרון שלו הוא שכאשר הheap הוא קטן התאים גם כן קטנים ולכן יצטרך לבדוק את רובם כי הם יתמלאו במהירות, מה שיכול לעקב את זמן הריצה במקום פשוט לרוץ על כל הheap פעם אחת כמו באלו שראינו בכיתה.
וכמובן כמו כל מנגנון איסוף זבל יש לו יתרון על פני שיחרור זבל ידני בכך שהוא לא עושה טעיות כמו שמתכנת יכול לעשות. וחיסרון בכך שצורך יותר זמן ריצה אם לדוגמא יש רק מעט זבל לשחרר ודבר זה ניתן לעשות במספר שורות בודדות.

# שאלה 3

1.ההבדל הוא שtype error זה שגיאה שנקבל בקוד כשמנסים לבצע פעולה שהיא לא חוקית על טיפוס מסוים וpseudo type error נקבל כשנבצע פעולה לא חוקית שהיינו מצפים שמערכת הטיפוסים תתפוס, אבל היא לא נתפסה בגלל מגבלות של הקומפיילר.
דוגמא לtype error:

```
void a(float b){}
int c=5;
a(c)
```

דוגנא לpseudo type error:

```
int x=5;
x/0;
```

2. שיטות השערוך:

שערוך להוט- שערוך שמבוצע לכל הפרמטרים של הפונקציה לפני תחילת הריצה של הפונקציה, לדוגמה בC

```
int func(int a){
return a;
}
func(1+1);
```

כשפעיל את הפונקציה func(1+1) אז בעצם יופעל func(2) בגלל שמתבצע קודם שיערוך לפרמטרים ורק אז קריאה לפונקציה.

שערוך נורמלי-השיערוך של הארגומנטים לפונקציה יעשה בכל פעם שנעשה בהם שימוש, דוכמא לכך בC תהיה מאקרו, לדוגמא המקרו הבא:

```
#define F(a,b) (a+b+a)
F(1+2,2+3);
```

אז מה שיקרה זה שבגלל שזה שיערוך נורמלי נשערך את 1+2 ואז את 2+3 ואז שוב השפה תשערך את 1+2 .

שערך עצל- שיערוך שבו משערכים כל ביטוי בפעם הראשונה בה משתמשים בו.דוגמא בC:

```
int x=f()+1;
x+1;
```

בביטוי הראשון x לא ישוערך אלא ישמר הביטוי הלא משוערך ורק כאשר נשתמש בx אחר כך אז הערך שלו ישוערך.

שערוך קצר- כאשר יש שערוך שנעשה בשלבים, אם אחד השלבים יקבע את הערך כולו אז לא נעשה שיערוך לשאר הביטוי, לדוגמא בC:

```
bool a= true || 2>1;
```

בביטוי הזה השיערוך קצר ולכן הוא יפסק ישר אחרי השלב הראשון בגלל שאם בOR אחד הרכיבים הוא TRUE אז הערך הכולל גם TRUE ולכן בכלל לא יתבצע שיערוך לביטוי 2>1.

3.בנאי אורתוגונלי הוא בנאי שניתן להפעיל אותו על כל טיפוס שקיים בשפה.