חלק יבש סימולציה 3

שאלה 1

סעיף א

שגיאה בשורה 13:

1) הטעות - n לא מוגדר בפונקציה
2) תיקון - כדי לגשת לשדה n של b יש לכתוב b.n

שגיאה בשורה 22:

1) הטעות - האופרטור "<" לא מוגדר עבור ארגומנט שמאלי שהוא const.
2) תיקון - יש לשנות את הגדרת הפונקציה בשורה 16 , להוסיף const בסוף שורת ההגדרה באופן הבא:

```
    bool operator <(const B& rhs) const{//added const

        return n < rhs.n;

}
```

שגיאה בשורה 30 :

1) הטעות- (b2+b3) הוא ערך זמני ומועבר לאופרטור + כרפרנס (הארגומנט הימני באופרטור + הוא רפרנס כפי שהוגדר).
2) תיקון - ניתן לפתור את הבעיה בכמה דרכים, אחת מהן היא שאופרטור + יקבל אובייקט ממש ולא רפרנס בארגומנט הימני , בצורה הבאה:

```
    B operator +(B b) {
```

סעיף ב

התוכנית מתחילה מmain כרגיל.

Main:

שורה 1:

יוצרת איבר מטיפוס B שנשמר בפוינטר לאיבר מטיפוס A. לA ולB לא מוגדרים constructors, ולכן יווצרו ויקראו default constructors. B יורש מA ולכן ייקרא קודם כל בנאי ברירת המחדל של A, ואז בנאי ברירת המחדל של B.

שורה 2:

הדפסה רגילה

"applying function f:"

שורה 3:

f(*pa) נקראת

בהעברת הארגומנטים לפונקציה של A מתבצע slicing על הB ש pa מצביע אליו, הוא יזוהה כA.

מתבצעת העתקה של הטיפוס החתוך ולכן נקראת הcopy constructor של A ושם מודפס לערוץ הפלט:

"A copy ctor"

לאחר העברת הארגומנט , מתבצעות פקודות הפונקציה f :

שורה 1, פונקציה f:

a.type() רואה את a כA ולכן יודפס:

"This is A"

לפי הנלמד בקורס, בשלב זה כיוון שהשורה הבאה היא return a ובה מתבצעת יציאה מהפונקציה, אמורה להתבצע הריסת עותק הארגומנט a שנוצר בכניסה לפונקציה. עם זאת, בעקבות שינויים בתקן c++ אין זה תמיד המצב, וסדר הפעולות עשוי להתנהל בצורה הבאה:

פונקציית main:

שורה 3:

בחזרה מהפונקציה מתבצעת העתקה של טיפוס מסוג A, שהוא ערך ההחזרה של פונקציית f. לכן הcopy constructor של A נקרא ומדפיס:

"A copy ctor"

על האויבקט שחזר מהפונקציה, מסוג A , מתבצעת המתודה type() ולכן מודפס:

"This is A"

כעת נהרסים שני אובייקטים מטיפוס A:

העותק של הארגומנט a שנוצר בכניסה לפונקציה f, לפי שינויי התקן שצוינו למעלה, וערך החזרה מפונקציה f שלא נשמר באף משתנה ולכן נהרס בסיום השורה. לכן מודפס:

"A dtor"

"A dtor"

שורה 4:

מדפיסה

":applying function g"

שורה 5:

הטיפוס אליו pa מצביע מועבר כרפרנס לפונקציה g (לא מתבצעת ההעתקה):

בפונקציה g :

כיוון שהגישה היא כרפרנס, a.type() קוראת למתודה type() שלB (לא מתבצע slicing בניגוד למקרה הקודם) ומודפס :

"This is B"

הערך החוזר מg הוא גם רפרנס ולכן לא מתבצעת ההעתקה בחזרה (שוב, בניגוד למקרה הקודם).

על האובייקט שחזר מהפונקציה g מתבצעת מתודת type(). כיוון שאובייקט זה הוא מסוג B ונשמר ברפרנס לטיפוס A ללא slicing, וכיוון שהפונקציה type() היא פונקציה וירטואלית ב Aשנדרסת בB, תיקרא הפונקציה type() כפי שהיא מוגדרת במחלקה B ויודפס:

"This is B"

Main:

שורה 6:

לבסוף, נמחק האובייקט שמוחזק ע"י pa. כיוון שpa הוא פוינטר לטיפוס A שמחזיק אובייקט מטיפוס B והורס הוא תמיד וירטואלי, קודם כל מתבצע התוכן של הdtor של B , ולאחר מכן התוכן של הdtor של A ומודפסות השורות:

""B dtor

"A dtor"

לסיכום , יודפסו השורות :

applying function f:

A copy ctor

This is A

A copy ctor

This is A

A dtor

A dtor

applying function g:

This is B

This is B

B dtor

A dtor

שאלה 2

סעיף א

```
class Car {
    public:
    Car()=default;
    virtual double getFuelConsumption(int speed) const=0;
}
```

מתודה אחת לפחות במחלקה Car המוגדרת ב pure virtual גורמת למחלקה להיות אבסטרקטית , ומחייבת את כל המחלקות היורשות ממנה לממש את הפונקציה.
במקרה שלנו זאת הפונקציה getFuelConsumption.

סעיף ב

```
double getPetrol(std::vector<Road> roads,const Car& car){
    double sum=0;
    for (Road i : roads){
        sum+=(i.length())/car.getFuelConsumption(i.speed());
    }
    return sum;
}
```