פתרון חלק יבש

שאלה 1:

סעיף א':

- שורה 13:

השגיאה היא שמנסים לגשת לאיבר פנימי של המחלקה B, אך ברגע שמגדירים את האופרטור להיות friend של המחלקה, הפונקציה לא מקבלת את this* כפרמטר לפונקציה.

אם לא היינו מגדירים את האופרטור כ friend של המחלקה, אך עדיין מכריזים עליו כמתודה של המחלקה - האופרטור היה מצפה לקבל את this* מצד שמאל לאופרטור, אך מאחר ששימוש זה הוא שימוש שגוי באופרטור, ובנוסף האופרטור הוא בינארי ולכן בצורת הכרזה זו, הוא יקבל גם את this*, ostream &out ו- const B& b – שזה יותר מדי פרמטרים.
לכן, לא נוכל להעמיס את האופרטור הנ"ל כמתודה למחלקה ונצטרך להעמיסו בתור friend.

כדי לתקן את שגיאה זו – נצטרך לשנות את שורה 13 לשורה הבאה:

```
out << "B: " << b.n;
```

- שורה 22:

השגיאה נובעת מניסיון לשלוח אובייקט מסוג const B& לאופרטור < שמוגדר כמתודה של המחלקה, שלא מקבל אובייקט מסוג const – ולכן עלול לשנותו.

כדי לתקן את השגיאה, נצטרך לשנות את שורה 16 לשורה הבאה:

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

- שורה 30:

השגיאה נובעת מנסיון לשלוח rvalue לאופרטור + שמוגדר לקבל lvalue reference.

נשים לב כי הצורה השקולה לכתוב את אגף שמאל של שורה 30 היא:

```
b1.operator+((b2 + b3))
```

מכאן ניתן לראות בבירור כי אנחנו מנסים לשלוח לאופרטור + את הביטוי b2+b3 שהוא rvalue.

כדי לתקן את השגיאה הנ"ל, ניתן להגדיר את האופרטור + כך שיקבל const B& rhs, ובאופן זה, הארגומנט שמקבל האופרטור יכול להיות גם מסוג rvalue, מכיוון שכך מובטח שהביטוי לא יעבור שינויים בלתי צפויים. בנוסף, כדי להבטיח שהאופרטור לא ישנה את האיברים הפנימיים של המחלקה B, נוסיף גם את המילה השמורה const לאחר סגירת הסוגריים.
לכן, נשנה את השורה 30 באופן הבא:

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

<u>סעיף ב':</u>

התוכנית תדפיס את סדרת הפלטים הבאה:

1. applying function f:
2. A copy ctor
3. This is A
4. A copy ctor
5. This is A
6. A dtor
7. A dtor
8. applying function g:
9. This is B
10. This is B
11. B dtor
12. A dtor

ניתן הסבר לכל שורה:

1. פלט קבוע של התוכנית.
2. הפונקציה f מצפה לקבל כקלט אובייקט מסוג A by value ולכן יקרא ה-copy ctor עם pa* כאובייקט שמועתק לאובייקט חדש כדי להעביר את האובייקט החדש כארגומנט לפונקציה f ("סלייסינג").
3. סוג האובייקט החדש a מוכר כאובייקט מהמחלקה A בתוך הפונקציה, ולכן תקרא הפונקציה type() מהמחלקה A.
4. כשנגיע בפונקציה f לשורת ה-return a, מאחר ו-f מחזירה אובייקט A, ואילו האובייקט a מוגדר רק בתוך הסקופ של הפונקציה f, נעתיק את a לתוך אובייקט זמני שנקרא לו לצורך ההסבר a_copy. אובייקט זה הוא מסוג A והוא נוצר על ידי קריאה ל-copy ctor של A.
5. הפונקציה type() בתוך הסקופ של main תיקרא עבור האובייקט הזמני a_copy שהוחזר מהפונקציה. מאחר וa_copy הוא אובייקט מהמחלקה A, תקרא הפונקציה type() מהמחלקה A.
6. קריאה לdtor של a_copy – מאחר וזהו אובייקט זמני שלא נשמר באף משתנה.
7. קריאה לdtor של a – מאחר ויצאנו מהסקופ של הפונקציה f.
8. פלט קבוע של התוכנית.
9. הפונקציה g מצפה לקבל const A& (לפי רפרנס). מאחר והמצביע pa מצביע לאובייקט מהמחלקה B – שהוא בן של A, כשנעביר את pa* לפונקציה g, היא תועבר לפי רפרנס בתור אובייקט ממחלקה B עם שם נרדף a. מאחר ו a הוא אובייקט מהמחלקה B, תיקרא הפונקציה type() של המחלקה B (כי type היא פונקציה וירטואלית)
10. a יוחזר לפי רפרנס מהפונקציה (כלומר, האובייקט pa* יוחזר מהפונקציה) ומאחר ו a הוא אובייקט מהמחלקה B, תיקרא הפונקציה type() של המחלקה B. (שוב, כי type היא פונקציה וירטואלית)

11+12. כשנרצה למחוק את ההקצאה הדינמית שהקצינו, מאחר ו-pa מצביע על אובייקט מהמחלקה B – שיורשת מ-A, ומאחר וה-dtor של A מוגדר להיות וירטואלי, יקרא קודם ה-dtor של B, ולאחר מכן ה-dtor של A.

שאלה 2:

סעיף א':

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

סעיף ב':

```
double getPetrol(std::vector<Road> roads, const Car& car)
{
    double cost = 0;
    double fuel_consumption = 0;
    std::vector<Road>::iterator end_iterator= roads.end();
    for(std::vector<Road>::iterator iterator= roads.begin();
                        iterator != end_iterator; ++iterator)
    {
        fuel_consumption = car.getFuelConsumption(iterator->speed());
        if(fuel_consumption == 0) continue; //Assume fuel consumption as km/liter
                            is not 0 (unless its an electric car for some reason?)
        fuel_consumption = (1/fuel_consumption) * iterator->length();
        cost += fuel_consumption;
    }
    return cost;
}
```