**פתרון חלק תאורטי:**

**1.** כל מתודה שאנחנו מגדירים pure virtual חייבים לממש – **נכון,** כאשר אנו משתמשים בPure virtual זה אומר שהבן של אותו אובייקט חייב לממש את אותה מתודה, אחרת הבן היורש יהיה גם הוא אבסטרקטי כלומר לא נוכל לייצר אובייקטים מסוג המחלקה שלו.

**2.** הסבירו מה ההבדל בין upcasting לבין slicing.

**Upcasting-** המרת מצביע\רפרנס בהורשה, כאשר מצביע לאב יכול להצביע על הבן (כל בן הוא מסוג של האב, למשל כל חתול הוא מסוג חיה). בהפעלת מתודות ווירטואליות מתנהג כבן, בהפעלת מתודות רגילות מתנהג כאב. Animal\* ptr = new Cat();

**Slicing-** חיתוך (חותכים מהבן את כל מה שלא שייך לאב) – המרת אובייקט בן לאובייקט האב. בסוף התהליך נקבל את אובייקט האב. Animal obj = new Cat();

**3.** לפנינו הורשה מסוג "יהלום" מחלקת האב A ממנה יורשים B,C ומהם יורשת המחלקה D. בהורשה מסוג זה נשתמש בהורשת וירטואלית כמו שמוצג בקוד הנתון.

השגיאות בקוד הן:

א. מחלקת האב A היא מחלקה אבסטרקטית (הורס-pure virtual), לא קיים לה בנאי דיפולטיבי לכן נוסיף לה – A() : f\_A(0) {};

ב. נקבל שגיאת קומפילציה משום שבבנאי של המחלקה D, בשורת האיתחול קוראים לבנאי של B וגם לבנאי של C אך לא קוראים לבנאי של A. לכן נקבל שגיאת קומפילציה במקרה כזה, על מנת לתקן אותו פשוט נוסיף קריאה לבנאי של A בשורת האיתחול של המחלקה D –

D(int num = 0, string str = "") : B(num, str), C(num, str) ,A(num) { f\_D = 0; }

**4.**

**א.**

|  |  |  |
| --- | --- | --- |
| **C class VTABLE** | **B class VTABLE** | **A class VTABLE** |
| C::~C() | B::~B() | A::~A() |
| B::f2() | B::f2() | A::f2() |
| B::f3() | B::f3() |  |
| C::f4() |  |  |

**ב.**

**שורה 1:** A::A()

**שורה 2:**B::B()

**שורה 3:** C::C()

**שורה 4:** i

**שורה 5:** f

**שורה 6:** a

**שורה 7:** e

**שורה 8:** e

**שורה 9:** A::A()

**שורה 10:** B::Copy Ctor

**שורה 11:** C::Copy Ctor

**שורה 12:** f

**שורה 13:** a

**שורה 14:** e

**שורה 15:** d

**שורה 16:** g

**שורה 17:** e

**שורה 18:** C::~C()

**שורה 19:** B::~B()

**שורה 20:** A::~A()

**שורה 21:** C::~C()

**שורה 22:** B::~B()

**שורה 23:** A::~A()

**מאור עטר - 318301231**