**עבודה הגשה מס'3 חלק ב'- חלק תיאורטי.**

**שם המגיש:** שגיא ביטון.

**תעודת זהות:** 205620859

1. התשובה היא **נכון.** הסיבה לכך היא שכאשר, מחלקת הבן יורשת ממחלקת האב Pure Virtual , אז בעצם קיימות בפני הבן 2 אופציות: 1. לממש שיטות אלי ולכן, מחלקה זו לא תהיה אבסקטרית, או לא לממש שיטות אלה ולהשאיר אותם גם Pure virtual ואז להפוך את המחלקה ל-Absctract Class בעצמה. לכן אם מחלקה יורשת אינה תממש pure virtual אז היא תהיה גם אבסטרקטית בעצמה.

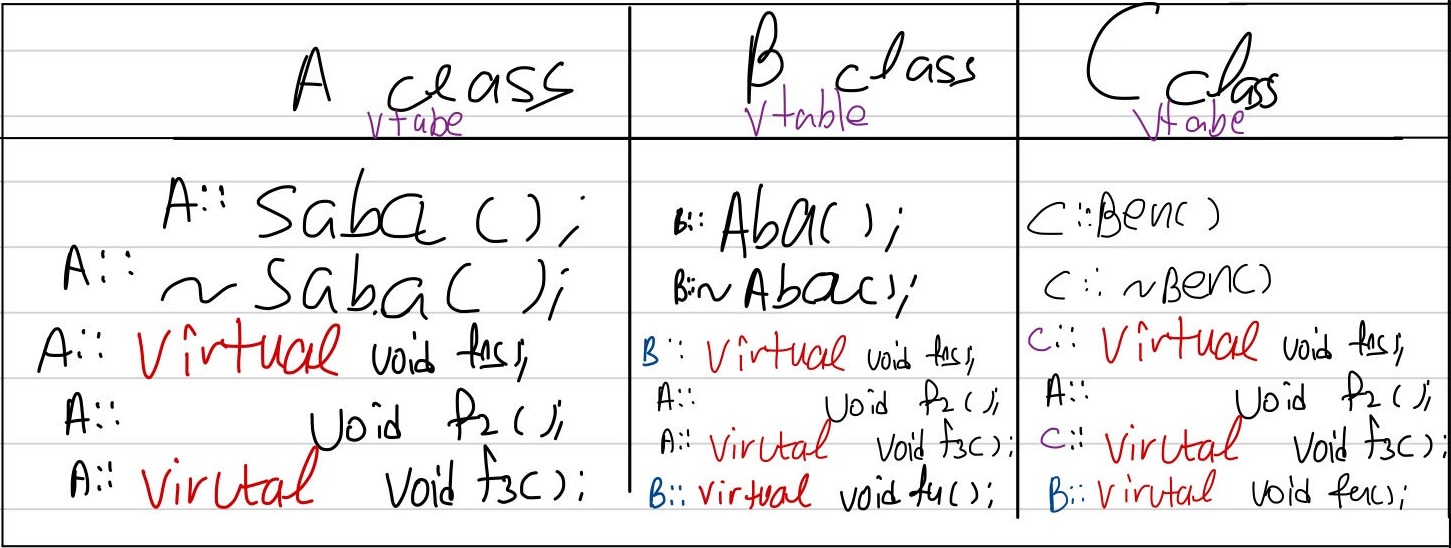
2. במידה ולא נרצה לאפשר יצירת אובייקטים מסוג B אלא רק מסוג C, עלינו להפוך את מחלקה B למחלקה אבקטרטית. אפשר לעשות זאת באמצעות מספר שיטות, אחת השיטות היא לא לממש מתודה ממחלקת האב במחלקת B ואז מחלקת B תהפוך למחלקה אבסזטקרטית בעצמה, כלומר לא נוכל ליצור אובייקטים. נוכל גם פשוט ליצור מתודה במחלקה B ולהגיד אותה Pure Virtual, ואז מחלקה זו תהפוך למחלקה מופשטת , כלומר לא ניתן ליצור אובייקטים מסוג מחלקה זו. אם נרצה ליצור אובייקטים ממחלקה C נממש את כל המתודות הפיור וירטואל שנמצאות במחלקה B וכך נדאג לכך שנוכל ליצור אובייקט מסוג C אך לא מסוג B.

3. בקוד קיימות 3 שגיאות. 1. במחלקת האב A מוגדרת פונקציה שהיא Pure Virtual לכן כל מחלקה יורשת, מחיובת לממש אותה או לעשות אותה גם Pure אצלה. כפי שניתן לראות , פונקציה הפיור שנמצאת במחלקה A היא Func1 , לכן התנאים חלים עליה במחלקות B ו-C. ניתן לראות את אותו שם מחלקה ב-B כביכול ממומש, אך פונקציה זו אינה זהה לחתימה של הפוקנציה במחלקה A חסר לה Const מיד () של הפונקציה על מנת לענות על הדרישה שהיא ממש את פונקצית ה-פיור.

2. שגיאה שניה היא כאשר אנו מנסים לעשות את השורה C=a1, הסיבות לכך הם שאופרטור = אינו ממומש באף אחת ממחלקות האלה, ולכן לא ניתן להשתמש בו מבלי לממש אופטרור שווה, מה עוד שמדובר בטיפוסים שונים.

3. שגיאה שלישית היא כאשר אנו מנסים לעשות C->SetX(1);

הסיבה לכך היא שאנו מנסים להשתמש במתודה זו, אך מתודה זו מוגדרת במחלקה A כמתודה אשר מקבלת משתנה מסוג int& כלומר, היא מצפה לקבל רפרנס למשתנה מסוג אינט, כלומר אנו מחוייבים לשלוח לה משתנה מסוג int ולא קבוע. זה לא תואם את מה שהיא אמורה לקבל.

4. 

5א. הפלט של קטע הקוד יהיה:

A::A()

B::B()

C::C()

A::f2()

C::f3()

c::f1()

A::f2()

C::f3()

B::~B()

A::~A()

דרך אגב הסיבה לכך שאנחנו לא רואים C::~C() היא שD'tor ב-A אינו מוגדר וירטואלית, לכן מתי שאני מגיעים לשורת delete הוא מוחק את טיפוס הזיכרון שביקשנו למחוק ולא הערך עצמו, לכן הוא ימחוק בסדר הפוך קודם את הבן ואז את האב ללא הילד! (כלומר A ואז B ללא C).לכן כ-ככל.. תמיד נגדיר דיסטקטור כוירטואל, ואז הוא ימחק קודם את הערך עצמו שהוא C ואז את B ואז את A.

ב. בגלל של f1() היא פונקציה וירטואלית במחלקת האב (A), הוא דוחה את הקומייפלר ומעביר את הקישור לזמן ריצה, ואז נפתחת טבלה הוירטואלית והוא נכנס לפונקציה המתאימה.

לגבי f2() נעשה קישור סטטי, כ- f2() היא לא פונקציה וירטואלית ועל כן הוא יכנס לפונקציה לפי ה-Class המתאם בזמן קומפלציה כלומר במקרה הזה A::void f2() כי זה הפונקציה היחידה שC ירשה B ו-B ירשה מ-A. ואז פונקציה זו אשר מגיעה ומפעילה אותה ממחלקה A מפעליה את אותו אובייקט עם פונקציה f3() שהיא אכן וירטואלית במחלקה C ועל כן, גם פה נעשה קישור דינמי ואובייקט נכנס למתודה הנכונה שמקרה הזה הוא נמצא במחלקה C(מסתכל על הטבלה הוירטואלית) .