

תכנות מערכות - עבודת הגשה 4

שאלה 1 :

(1) הפלט התוכנית הנתונה :

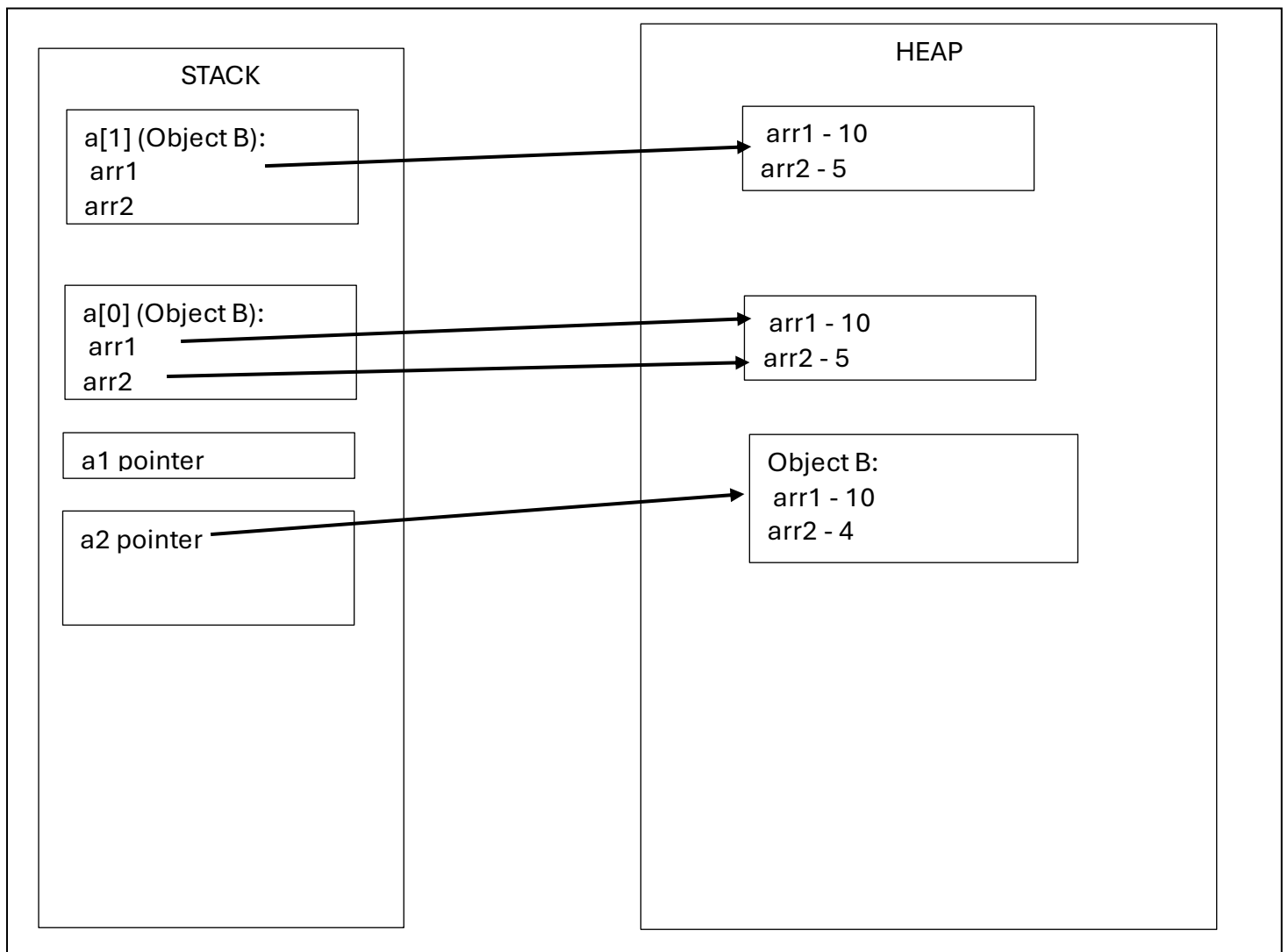
in constructor A
in constructorr B
in constructor A
in constructorr B
in constructor A
in constructorr B
in f1 A
in dtor B
in dtor A
in constructor A
in constructorr B
in f2 B
in f1 B
in dtor B
in dtor A
in dtor B
in dtor A

```
int main(){  
    B a[2]; → יצירה של מערך בגודל 2 של אובייקטים מסוג B  
    A *a1 = new B() → קריאה לבנאים של A של B  
    a1->f1(3); הפעלה של פונקציה A::F1  
    delete a1; → מחיקת האובייקט הפוך מסדר היצירה  
    A *a2 = new B(); → יצירה של אובייקט חדש מסוג B  
    a2->f2(4); → F2 וירטואלית , תופעל הפונקציה של B::F2  
    a[0].f1(5); → קריאה לפונקציה B::F1 על האובייקט הראשון במערך  
    return 0;  
}
```

(2) בקוד הנתון יש מספר זליגות זיכרון :

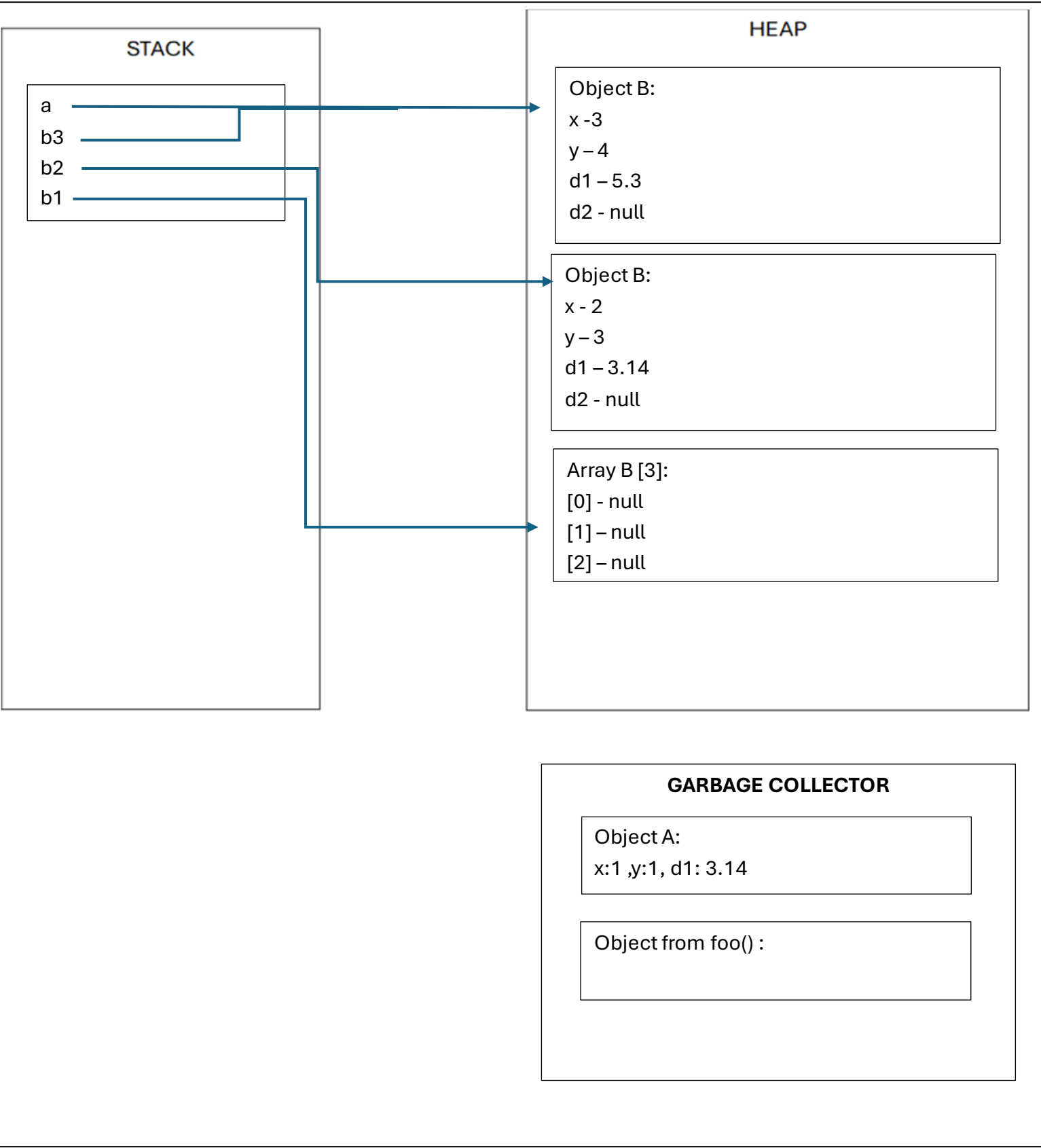
- א. זליגה ב-A::f1- הפונקציה מבצעת `arr2 = new int[size]` אם כבר היה זיכרון מוקצה ב- `arr2` הכתובת הקודמת נמחקת.
- ב. זליגה ב `B::f2`, `B::f1`, `A::f1` - כולן מבצעות הקצאה מחדש ל `arr2` / `arr1` מבלי לבצע שחרור זיכרון קיים בכל אחד מהם.
- ג. שימוש ב `new` ב `a2` אבל לא בוצעה מחיקה עם `.delete`.
- ד. ההורס של A משחרר רק את `arr2` וההורס של B משחרר רק את `arr1`, במצב כזה שפונקציה מקצה אובייקט לשניהם תהיה זליגה כי כל מחלקה אחראית על מצביע אחד בלבד.

(3) תמונת זיכרון :



שאלה 2 :

(א



(ב) ה garbage collector ימחק את כל האובייקטים ב heap שאין להם מצביע קיים ב stack ,
האובייקטים שימחקו הם :

1. האובייקטים שנוצרו בתוך פונקציית foo() :
2. בתוך foo(a) נוצר אובייקט חדש וברגע שהפונקציה הסתיימה המשתנה המקומי a נמחק והאובייקט נשאר ללא מצביע קיים.
2. בתוך foo(b2, b1[1]) ,נוצרו אובייקט new A() ואובייקט new B() גם הם נשארו ללא מצביעים ברגע שהפונקציה הסתיימה (כי a1 ו-a2 היו משתנים מקומיים).
3. אובייקט ה A-הראשון שנוצר בשורה הראשונה של ה :main-ברגע שביצענו a = b3 בשורה האחרונה, איבדנו את המצביע היחיד שהיה לאובייקט ה A-המקורי.

שאלה 4 :

- תהליך ה – Verification בשלב ה – Linking :
- שלב זה הוא חלק מהתהליך ה – Linking , ומטרתו העיקרית היא להבטיח שקובץ ה – ByteCode שנטען הוא בטוח להרצה ואינו מפר את כלל ה- JVM.
- מאחר שקובץ מחלקה יכול להגיעו ממקורות לא ידועים ה- JVM אינה סומכת על הקוד ומבצעת סדרת בדיקות :
1. בדיקת תקינות מבנית - וידוא שהקובץ בנוי בפורמט נכון של JAVA ושהכל במבנה חוקי.
 2. אימות ByteCode - זהו החלק המורכב ביותר, בו המערכת עוברת על פקודות הקוד ומוודאת שהן "הגיוניות". לדוגמה:
 - שאין חריגה ממחסנית. (Stack Overflow/Underflow)
 - שסוגי הנתונים מתאימים לפעולות המבוצעות עליהם.
 - שכל פקודת קפיצה בתוך הקוד מובילה לכתובת חוקית בתוך אותה מתודה.
 3. בדיקת הרשאות - וידוא שהקוד לא מנסה לבצע פעולות אסורות, כמו גישה לשדות פרטיים של מחלקה אחרת או ירושה ממחלקה שהוגדרה כ- final
 4. אתחול משתנים - וידוא שכל משתנה מקומי מקבל ערך לפני שמשתמשים בו בפועל.