

פתרון מועד א' בסדנת תכנות ב - C++

(תאריך הבחינה : 1.2.05)

1. (6%)

נתון הקוד הבא :

```
#include <iostream>

int main() {
    int *p = new int(7), **pp = &p;
    int &a1 = **pp;
    int *&a2 = *pp;
    int **&a3 = pp;
    a1++; a2++; a3++;
    std::cout << a1 << "\n";
    delete &a1;
}
```

בחרו בתשובה הנכונה :

- א. הקוד תקין והפלט שלו הוא 8.
- ב. פלט התוכנית אינו מוגדר.
- ג. הקוד לא משחרר את כל הזיכרון שהוקצה.
- ד. הקוד תקין והפלט שלו הוא 7.

תשובה: א.

נתון הקוד הבא (מתייחס לשאלות 2 ו 3) :

```
#include <iostream>

struct A {
    A(char c) { std::cout << c << ' '; }
};

struct B : public A {
    A _a;
    B *_p;
    B(char c1, char c2) : A(c1), _a(c2), _p(0) {}
    B(char c1, char c2, char c3) : A(c1), _a(c2), _p(new B(c2,c3)) {}
    ~B() { if(_p) delete _p; }
};

int main() {
    B b('a','b','c');
}
```

2 (6%) מה יהיה הפלט ? _____

תשובה: a b b c

3 (3%)

האם יש זליגת זיכרון (זיכרון שלא שוחרר בסיום התוכנית) _____ (ענו רק "כן" או "לא")

תשובה: לא.

4 (6%)

מה יהיה הפלט של הקוד הבא:

```
#include <iostream>
class A {
    int _a;
public:
    static A theA;
    A(int a) : _a(a) {}
    A(const A& other) { std::cout << other._a << " "; }
    void foo(A other=theA) {}
};

A A::theA(17);

int main() {
    A a(5);
    a.foo();
    A::theA.foo(a);
}
```

פלט: _____

תשובה: 17 5

5 (10%)

נתונה המחלקה הבאה:

```
class Array3d{
    int*** _array;
    int _x,_y,_z;
public:
    Array3d(int x, int y, int z):_array(new int**[x],_x(x),_y(y),_z(z) {
        for(int i=0; i<x; ++i) {
            _array[i] = new int*[y];
            for(int j=0; j<y; ++j)
                _array[i][j] = new int[z];
        }
    }
    ....
};
```

המחלקה מייצגת מערך תלת ממדי. עליכם להוסיף למחלקה copy constructor תקין, המעתיק את כל תכולתה:

(ניתן לכתוב את הפונקציה כ- inline בתוך הצהרת המחלקה אפילו אם היא ארוכה מפונקציות inline רגילה)

תשובה:

```
Array3d(const Array3d& other) : _array(new int**[other._x],
                                     _x(other._x),_y(other._y),_z(other._z)) {
    for(int i=0; i<_x; ++i) {
        _array[i] = new int*[_y];
        for(int j=0; j<_y; ++j) {
            _array[i][j] = new int[_z];
            for(int k=0; k<_z; ++k)
                _array[i][j][k] = other._array[i][j][k];
        }
    }
}
```

6 (6%)

מה יהיה הפלט של הקוד הבא:

```
#include <iostream>
#include <cmath>

struct A {
    int operator()(int x) { return 2*x; }
};

template <typename Function>
void pf3(Function f) {
    std::cout << f(f(f(1))) << " ";
}

int main() {
    pf3(A());
    pf3(sqrt);
}
```

פלט: _____

תשובה: 8 1

7 (6%)

מה יהיה הפלט של הקוד הבא:

```
#include <iostream>

struct Base {
    Base *_p;
    virtual void foo() {_p->foo();}
};

struct A : public Base {};

struct B : public Base {
    virtual void foo();
};

struct C : public Base {
    virtual void foo() {}
};

A a1, a2;
B b;
C c;

void B::foo() {
    std::cout << "he he ";
    a2._p = &c;
    _p->foo();
}

int main() {
    a1._p = &a2;
    a2._p = &b;
    b._p = &a1;
    a1.foo();
}
```

פלט:

תשובה: he he

8 (5%)

כתבו מה יקרה אם נשנה את המימוש של הפונקציה C::foo() של שאלה 7, באופן הבא:

```
void C::foo() {
    _p = &b;
    _p->foo();
}
```

(הניחו שהפונקציה ממוקמת כך שהקוד עובר קומפילציה)

תשובה: רקורסיה אין סופית עם הדפסות של he he בלתי פוסקות.

9 (7%)

הסבירו מדוע מודל ה- templates הקיים, המצריך כתיבת מימושי פונקציות ב- headers, עלול לגרום לקובצי הרצה גדולים שלא לצורך.

תשובה: במודל הקיים, ייתכן מצב שבו אותה פונקצית template, תיפרש לבקשת שני קבצי cpp. שונים. מכיוון שלפונקציות האלה internal-linkage, ה- linker לא ינסה לבצע קישור בין קבצי ה- object השונים ויכלול בקובץ ההרצה את שני העותקים. ברור שקיום שני עותקים של אותו קוד בקובץ הריצה מהווה בזבוז.

10 (6%)

מה יהיה פלט התוכנית הבאה:

```
#include <iostream>
```

```
class A {};
```

```
A* operator -(A& a) {  
    static int count=0;  
    std::cout << ++count << " ";  
    return &a;  
}
```

```
int main() {  
    A a;  
    *-*-*-a;  
}
```

פלט: _____

תשובה: 1 2 3 4

נתון הקוד הבא (מתייחס לשאלות 11, 12 ו- 13):

```
template<typename T>  
struct A{T** _t};
```

```
template<typename T>  
void foo(A<A<T**> &a) {a._t;}
```

```
int main() {  
    A<A<A<char>***> > > c;  
    foo(c);  
}
```

11 (5%)

פרטו את כל הטיפוסים הנפרשים מה - template של A בתוכנית :

תשובה: A<char>, A<A<char>***>, A<A<A<char>***>> - ו

12 (5%)

לאיזה טיפוס יהיה שווה T בפריסה של הפונקציה foo: _____

תשובה: A<char>*

13 (6%)

מה יהיה הטיפוס של הביטוי "a._t" הנמצא בגוף הפונקציה foo: _____

תשובה: A<A<char>***>**

14 (6%)

נתון הקוד הבא :

```
template <typename T, int SIZE>
class Array {
    T _array[SIZE];
public:
    T& operator[](int i) { return _array[i]; }
};
```

```
int main() {
```

```
    _____
    for(int i=0; i<5; ++i)
        for(int j=0; j<7; ++j)
            arr2d[i][j] = i*j+0.5;
}
```

הוסיפו בשורה החסרה בקוד את ההגדרה של המשתנה arr2d כך שהקוד יהיה תקין. המשתנה אמור לתפקד כמערך דו ממדי בגודל 5x7. אין להשתמש במערכים רגילים אלא רק ב - template שהוגדר.

תשובה:

```
Array<Array<double,7>,5> arr2d;
```

15 (7%)

מה יהיה הפלט של הקוד הבא :

```
#include <iostream>
```

```
class Printable {
public:
    virtual void print() = 0;
};
```

```
class B : public Printable {
public:
    virtual void print() { std::cout << "B "; }
};
```

```
template <typename T>
class Pair : public Printable {
public:
    T *_first, *_second;
    virtual void print() {
        _first->print();
        _second->print();
    }
};
```

```
int main() {
    B b;
    Pair<B> p1;
    p1._first = p1._second = &b;
    Pair<Pair<B> > p2;
    p2._first = p2._second = &p1;
    Pair<Printable> p3;
    p3._first = &p1;
    p3._second = &p2;
    p3.print();
}
```

פלט:

תשובה:

B B B B B

16 (5%)

מה היה קורה אם בסוף ה- main של השאלה הקודמת, היינו מוסיפים את השורות:

```
p3._second = &p3;
p3.print();
```

- א. הקוד לא היה עובר קומפילציה.
- ב. רקורסיה ללא תנאי עצירה.
- ג. זליגת זיכרון: זיכרון שמוקצה ולא משוחרר.
- ד. התוכנית הייתה עובדת בצורה תקינה.

תשובה: ב.

17 (5%)

נתון הקוד הבא:

```
#include <iostream>
```

```
class A {
public:
    static int _c;
    void foo() {
        if(_c++==2)
            return;
```

```

    foo();
}
void foo() const {
    if(_c++==4)
        return;
    foo();
}
};

int A::_c=0;

int main() {
    A a;
    const A ca(a);
    a.foo();
    ca.foo();
    std::cout << A::_c << "\n";
}

```

בחרו את התשובה הנכונה :

- א. הקוד לא יעבור קומפילציה מכיוון שפונקציה קבוע מנסה לשנות משתנה של המחלקה.
- ב. התוכנית תכנס לרקורסיה אין סופית בזמן הריצה.
- ג. הקוד לא יעבור קומפילציה מכיוון שיש כפל משמעות באחת ההפעלות של foo.
- ד. התוכנית תרוץ והפלט יהיה 5.

תשובה : ד.

בהצלחה !