

**פתרון הבחינה**

פקולטה: מדעי הטבע

מחלקה: מדעי המחשב

שם הקורס: מבנה זיכרון ושפת ++C

קוד הקורס: 2-7027810 כל הקבוצות - קבוצות 1,3,4,5

מועד \_\_**א**\_\_\_ סמסטר: \_\_\_**ב**\_\_\_ שנה: ה'תשעט

תאריך הבחינה: ט"ו בסיון ה'תשע"ט, 18/06/2019

משך הבחינה: 3 שעות

שם המרצים: אראל סגל-הלוי, גיל בן-ארצי

# **שאלה 1 [9 נק']**

1. **#include** <iostream>
2. **using namespace** std;
3. **class** A {
4. **public**:
5. A() { cout << **"A(0)"** << endl; }
6. A(**const** A& a) { cout << **"A(1)"** << endl; }
7. };
8. **class** B {
9. **public**:
10. B() : a() { cout << **"B(0)"** << endl; }
11. B(**const** B& b) { cout << **"B(1)"** << endl; }
12. **private**:
13. A a;
14. };
15. **int** main() {
16. B object1;
17. B object2 = object1;
18. **return** 0;
19. }

א [3 נק']. מה פלט התוכנית?

*פתרון*:

A(0) B(0) A(0) B(1)

**בשורה 17** בונים עצם מסוג B בעזרת בנאי בלי ארגומנטים, המוגדר בשורה 10. לשם כך צריך קודם לבנות את השדה מסוג A, וזה נעשה בשורה 10 ע"י בנאי בלי ארגומנטים, המוגדר בשורה 5. לכן קודם מודפס A0 ואחר-כך B0.

**בשורה 18** בונים עצם מסוג B בעזרת בנאי מעתיק, המוגדר בשורה 11. לשם כך צריך קודם לבנות את השדה מסוג A. כיוון ששורת האתחול לא אומרת איך לבנות את השדה הזה, משתמשים בבנאי בלי ארגומנטים של A, המוגדר בשורה 5. לכן קודם מודפס A0 ואחר-כך B1.

ב [3 נק']. אם מוחקים את שורה 11 , מה פלט התוכנית?

*פתרון*:

A(0) B(0) A(1)

**בשורה 17** מודפס A0 ואחר-כך B0 בדיוק כמו קודם.

**בשורה 18** בונים עצם מסוג B בעזרת בנאי מעתיק, אבל אין בנאי מעתיק, ולכן קוראים לבנאי מעתיק ברירת-מחדל. ע"פ הגדרה, בנאי מעתיק ברירת-מחדל פשוט קורא לבנאי מעתיק של כל השדות. במקרה זה, יש רק שדה אחד מסוג A, ולכן קוראים לבנאי המעתיק שלו ומדפיסים A1.

ג [3 נק']. אם מחזירים את שורה 11 ומוחקים את שורה 6, מה פלט התוכנית?

*פתרון*:

A(0) B(0) A(0) B(1)

הפתרון זהה לסעיף א. העובדה שמחקנו את שורה 6 לא רלבנטית, כי במקרה זה אנחנו בכלל לא מגיעים לשורה 6 – ראו בהסבר של סעיף א. למרות שקוראים לבנאי מעתיק של B, הוא לא קורא לבנאי מעתיק של A אלא לבנאי בלי ארגומנטים.

# **שאלה 2 [9 נק']**

1. **#include** <iostream>
2. **using** **namespace** std;
3. **class** A;
4. **class** B {
5. **public**:
6. **explicit**B (A& o) { cout << "B (A)" << endl; }
7. };
8. **class** A {
9. **public**:
10. A() { cout << "A()" << endl; }
11. };
12. **void** printit(B arg) {}
13. **int** main() {
14. A a;
15. B b(a);
16. printit(a);
17. **return** 0;
18. }

א [3 נק']. בתוכנית יש שגיאת קומפילציה בשורה 21. הסבירו במילים שלכם מהי השגיאה ומדוע היא קורה.

*פתרון*: קוראים לפונקציה שמצפה לקבל B אבל מעבירים משתנה מסוג A (1 נק').

אמנם יש בנאי ממיר, אבל הוא מוגדר כ-explicit ולכן הקומפיילר לא קורא לו אוטומטית (2 נק').

ב [3 נק']. הציעו תיקון לשורה 21 כך שהתוכנית תתקמפל ותרוץ בלי שגיאות. אין לשנות שורות אחרות.

*הפתרון שרצינו שתכתבו*: לקרוא בפירוש לבנאי הממיר: printit(B(a)).

*פתרון נוסף שמתקבל כנכון*: printit(b).

ג [3 נק']. מהו פלט התוכנית לאחר התיקון?

*פתרון*: לפי הפתרון הראשון:

A(),B(A), B(A)

לפי הפתרון השני:

A(),B(A)

# **שאלה 3 [9 נק']**

1. #include **<iostream>**
2. **using namespace** std;
3. **class** Shape {
4. **public**:
5. **virtual void** draw() = 0;
6. **virtual** ~Shape() { cout << **"~Shape()"** << endl; }
7. };
8. **class** Circle : **public** Shape {
9. **public**:
10. **void** draw() { cout << **"Circle::draw()"** << endl; }
11. ~Circle() { cout << **"~Circle()"** << endl; }
12. };
13. **void** drawTheShape(Shape s) {
14. s.draw();
15. }
16. **int** main() {
17. Circle c;
18. drawTheShape(c);
19. **return** 0;
20. }

א [3 נק']. בתוכנית יש שגיאת קומפילציה בשורה 19. הסבירו במילים שלכם מהי השגיאה ומדוע היא קורה.

*פתרון*: כיוון שמעבירים את הפרמטר by value, אמור להיווצר משתנה חדש מסוג Shape בכניסה לפונקציה, אבל המחלקה היא וירטואלית-טהורה ולכן אי-אפשר ליצור משתנה ממחלקה זו.

ב [3 נק']. הציעו תיקון לשורה 19 כך שהתוכנית תתקמפל ותרוץ בלי שגיאות. אין לשנות שורות אחרות.

*הפתרון שרצינו שתכתבו*: להעביר את הפרמטר by reference, כך:

void drawTheShape(Shape& s).

במקרה זה רק כתובת המשתנה מועברת, ולא נוצר משתנה חדש.

*פתרון אחר שמתקבל כנכון:*

void drawTheShape(Circle s).

ג [3 נק']. מהו פלט התוכנית לאחר התיקון?

*לפי הפתרון הראשון לסעיף ב:*

Circle::draw()

~Circle()

~Shape()

*לפי הפתרון השני לסעיף ב:*

Circle::draw()

~Circle()

~Shape()

~Circle()

~Shape()

# **שאלה 4 [9 נק']**

נתונה התוכנית:

01 #include <iostream>

02 #include <string>

03 using namespace std;

04

05 class string {

06     const char\* p;

07 public:

08     string(const char\* p) {this->p = p;}

09 };

10

11 int main() { // a demo program

12     string s = "abc";

13 }

כשמריצים make מתקבלת ההודעה הבאה:

main.cpp:12:2: error: reference to 'string' is ambiguous

string s = "abc";

^

א [3 נק']. מה סוג השגיאה (קומפילציה / קישור / זמן-ריצה / לוגית / דליפת זיכרון)? מה בדיוק גורם לשגיאה?

*פתרון*: שגיאת קומפילציה. היחידות iostream, string כוללות את std::string (1 נק'), וכשמבצעים using namespace std המחלקה עוברת למרחב הראשי (2 נק'). בשורה 5 מוגדרת מחלקה אחרת בשם string. לכן בשורה 12 הקומפיילר לא יודע לאיזה string אנחנו מתכוונים, וצועק שהשורה היא רב-משמעית – ambiguous.

ב [3 נק']. איך אפשר לתקן את הבאג ע"י **מחיקה** של שורה אחת? אין לשנות את התוכנית הראשית.

*פתרון*: למחוק את שורה 3. ואז המחלקה התקנית string תישאר במרחב השם std ולא תתנגש עם ה-string שאנחנו מגדירים (לשם כך נועד מרחב-שם).

* הערה: הפתרון של מחיקת שורה 2 לא מספיק, כי string מוגדר גם בiostream. הוא מזכה ב-2 מתוך 3 נקודות.

ג [3 נק']. במחלקה הנתונה יש פוינטר p אבל אין מפרק (destructor). האם זה עלול לגרום לשגיאה כלשהי? אם כן, מה סוג השגיאה שעלולה להיווצר? אם לא, מדוע לא?

*פתרון*: זו לא שגיאה, אין צורך לפרק שום דבר כי הבנאי של string לא בונה שום דבר. הוא מקבל את הפוינטר מבחוץ. המחיקה צריכה להתבצע ע"י אותו גורם שיצר את הפוינטר ע"י new.

# **שאלה 5 [22 נק']**

סטודנט כתב תבנית-מחלקה (class template) בשם Tree המייצגת עץ-מיון בינארי.

כיתבו בדיקות-יחידה מקיפות עבור המחלקה. יש לבדוק את הדרישות הבאות:

* אפשר ליצור עץ בינארי עם כל סוג שיש לו אופרטור "קטן מ-" (<).
* העץ אמור לשמור על סדר הפריטים בתוכו, בהתאם לאופרטור "קטן מ-" שלהם.
* העץ אמור לאפשר הכנסה (**insert**), וכן לבדוק מי הבן השמאלי (**left**) של פרט כלשהו בעץ.

א [11 נק']. **תיכנון הבדיקות**: כיתבו (בעברית) רשימה מפורטת של מקרים שאתם מתכוונים לבדוק.

*פתרון*: יש לבדוק לפחות שלושה עצים – שניים מהם יכולים להיות העצים הנתונים (מחרוזת ומספר), ועץ שלישי צריך להיות עם מחלקה חדשה (למשל Person) שיש לה אופרטור "קטן מ-".

כמו כן, יש להכניס לכל עץ לפחות 5 עצמים ולבדוק לפחות שלושה מצבים, כגון:

* עץ ימני בלבד
* עץ שמאלי בלבד
* עץ מאוזן
* וכד'..

ב [11 נק']. **מימוש הבדיקות**: מצורף קובץ עם בדיקות לדוגמה. עליכם להרחיב אותו ולהוסיף לו את הבדיקות בהתאם לתכנון שכתבתם בסעיף א. הפונקציות CHECK\_OK, CHECK\_EQUAL , CHECK\_THROWS כבר מוגדרות בקובץ badkan.hpp, כמו במטלות. אין צורך לממש אותן. מותר להוסיף, לצורך הבדיקות, מחלקות ו/או פונקציות לפני התוכנית הראשית.

פתרון אפשרי בעמוד הבא.

בכל סעיף, מלוא הניקוד ניתן למי שבדק לפחות שלושה עצים, אחד מהם עם מחלקה חדשה עם אופרטור<, וכל עץ כולל לפחות 5 הכנסות הבודקות מקרים שונים.

**שימו לב**: יש סטודנטים שהציעו לבדוק עץ עם מחלקה **שאין** לה אופרטור "קטן מ", ולבדוק שזה גורם לחריגה. הבעיה היא שזה לא גורם לחריגה – זה גורם לשגיאת קומפילציה, כך שאי-אפשר לבדוק את זה ע"י CHECK\_THROWS.

בדומה לכך, יש סטודנטים שרצו לבדוק הכנסה של מספר לעץ של מחרוזת – גם זה יגרום לשגיאת קומפילציה ולא לחריגה.

#include <iostream>

#include "badkan.hpp"

#include "Tree.hpp"

using namespace std;

struct Person {

string name;

int age;

// compare by age

bool operator<(const Person& other) {

return age < other.age;

}

// for CHECK\_EQUAL

bool operator==(const Person& other) {

return age==other.age && name==other.name;

}

};

int main() {

  ariel::Tree<int> itree;

  ariel::Tree<string> stree;

  ariel::Tree<Person> ptree;

  badkan::TestCase testcase("Binary tree");

testcase

  .setname("Tree of string, left only")

  .CHECK\_OK (stree.insert("eee"))

  .CHECK\_OK (stree.insert("ddd"))

  .CHECK\_OK (stree.insert("ccc"))

  .CHECK\_OK (stree.insert("bbb"))

  .CHECK\_OK (stree.insert("aaa"))

  .CHECK\_OK (stree.left("eee"), string{"ddd"})

  .CHECK\_OK (stree.left("ddd"), string{"ccc"})

  .CHECK\_OK (stree.left("ccc"), string{"bbb"})

  .CHECK\_OK (stree.left("bbb"), string{"aaa"})

  .CHECK\_THROWS(stree.left("aaa"))

  .CHECK\_THROWS(stree.left("xxx"))

  ;

  testcase

  .setname("Tree of int, right only")

  .CHECK\_OK (itree.insert(1))

  .CHECK\_OK (itree.insert(2))

  .CHECK\_OK (itree.insert(3))

  .CHECK\_OK (itree.insert(4))

  .CHECK\_OK (itree.insert(5))

  .CHECK\_THROWS(itree.left(1))

  .CHECK\_THROWS(itree.left(2))

  .CHECK\_THROWS(itree.left(3))

  .CHECK\_THROWS(itree.left(4))

  .CHECK\_THROWS(itree.left(5))

  .CHECK\_THROWS(itree.left(6))

  ;

  Person pa{"a",55}, pb{"b",22}, pc{"c",88}, pd{"d",11}, pe{"e",33},

   pf{"f",77}, pg{"g",99};

testcase

  .setname("Balanced tree of custom class (Person), ordered by age")

  .CHECK\_OK (ptree.insert(pa))

  .CHECK\_OK (ptree.insert(pb))

  .CHECK\_OK (ptree.insert(pc))

  .CHECK\_OK (ptree.insert(pd))

  .CHECK\_OK (ptree.insert(pe))

  .CHECK\_OK (ptree.insert(pf))

  .CHECK\_OK (ptree.insert(pg))

  .CHECK\_EQUAL(ptree.left(pa), pb)

  .CHECK\_EQUAL(ptree.left(pb), pd)

  .CHECK\_EQUAL(ptree.left(pc), pf)

  .CHECK\_THROWS(ptree.left(pd))

  .CHECK\_THROWS(ptree.left(pe))

  .CHECK\_THROWS(ptree.left(pf))

  .CHECK\_THROWS(ptree.left(pg))

  .print();

}

# **שאלה 6 [22 נק']**

הגדירו תבנית (template)של מחלקה בשם matrix המייצגת מטריצה. יש לממש את ה template בשם כך שניתן יהיה להריץ בעזרתו את התוכנית הראשית ללא שגיאות. המחלקה צריכה לתמוך בפעולות הבאות:

* איתחול (בניה) מתוך וקטור של וקטורים. חתימת הבנאי:

Matrix(const std::vector<std::vector<T>>& V);

* הדפסת מטריצה בעזרת אופרטור הפלט;
* חיבור שתי מטריצות באותו גודל בעזרת אופרטור +;
* כפל מטריצה בוקטור בעזרת אופרטור \*.

**הגדרת מכפלת מטריצה בוקטור**: נתונה מטריצה KxM ווקטור Mx1. המכפלה שלהם הינה מטריצה/וקטור בגודל Kx1. האיבר ה-j הינו המכפלה של השורה ה-j במטריצה בוקטור הנתון.

הערות:

* בנוסף לפונקציות הכתובות למעלה, יש לממש כל פונקציה נוספת הדרושה כדי שהתוכנית תעבוד.
* בכל מצב של קלט לא תקין, יש לזרוק חריגה.

**int** main() {  
 std::vector <std::vector <**int**> > Mat {{1,2,3},{4,5,6},{7,8,9}};  
 Matrix<**int**> a (Mat); // constructs a matrix from vector of vectors  
 Matrix<**int**> b ( {

{4, 5, 6},  
 {9, 0, 1},  
 {7, 8, 7} });  
 std::cout << **"a: \n"** << a;  
 std::cout << **"b: \n"** << b;  
 std::cout << **"a + b: \n"** << a+b; // should print:

// 5 7 9

// 13 5 7

// 14 16 16

Matrix<**int**> d ( {

{1, 2},  
 {4, 5},  
 {7, 8}});  
 std::vector<**int**> d\_vector = {1, 2};   
 std::cout << **"d \* d\_vector: \n"** << d \* d\_vector; // should print:

// 5

// 14

// 23

**return** 0;  
}

פתרון אפשרי בעמוד הבא. **מפתח ניקוד**:

סעיף א [11 נק']: כותרות נכונות לכל השיטות;.

סעיף ב [11 נק']: מימושים נכונים ויעילים לכל השיטות.

// Matrix.hpp

#include **<iostream>**#include **<vector>**#include **<algorithm>**#include **<iterator>**#include **<initializer\_list>  
  
template**<**typename** T> **class** Matrix;  
  
**template**<**typename** T>  
std::ostream& **operator**<<(std::ostream&, **const** Matrix<T>& );  
  
**template**<**typename** T>  
Matrix<T> **operator**+(**const** Matrix<T>& , **const** Matrix<T>& );  
  
**template**<**typename** T>  
Matrix<T> **operator**\*(**const** Matrix<T>& , **const** std::vector<T>& );  
  
**template**<**typename** T>  
Matrix<T> **operator**\*(**const** Matrix<T>& , **const** Matrix<T>& );  
  
*// Class definition***template**<**typename** T>  
**class** Matrix {  
 **friend** std::ostream& **operator**<< <>(std::ostream& , **const** Matrix<T>&);

**friend** Matrix **operator**+ <>(**const** Matrix& , **const** Matrix& );  
 **friend** Matrix **operator**\* <>(**const** Matrix& , **const** std::vector<T>& );  
 **friend** Matrix **operator**\* <>(**const** Matrix& , **const** Matrix& );  
  
**public**:

Matrix(size\_t r, size\_t c);  
 Matrix(**const** std::vector<std::vector<T>>&);  
 Matrix& **operator**=(**const** Matrix& );  
  
**private**:  
 size\_t rows;  
 size\_t cols;  
 std::vector<std::vector<T>> data;  
};  
  
**template**<**typename** T>  
std::ostream& **operator**<<(std::ostream& out, **const** Matrix<T>& m) {  
 **for** (**const auto**& row : m.data) {  
 **for** (**const auto**& element : row)  
 out << element << **'\t'**;  
 out << **'\n'**;  
 }  
 **return** out;  
}

**template**<**typename** T>  
Matrix<T>::Matrix(size\_t r, size\_t c) : rows(r), cols(c) {

data.reserve(r);  
 **for** (**auto** row : data)  
 row.reserve(c);  
}

**template**< **typename** T>  
Matrix<T>::Matrix(**const** std::vector<std::vector<T>>& V){  
  
 rows=V.size();  
 cols=V.at(0).size();  
 data = V;}  
  
  
  
**template**<**typename** T>  
Matrix<T> **operator**+(**const** Matrix<T>& left, **const** Matrix<T>& right) {

if ((right.rows != left.rows) || (right.cols != left.cols)){

throw std::range\_error("matrix rows/cols is out of range");

}

Matrix<T> result (left.rows, left.cols);  
  
 std::vector<T> temp;  
  
 **for** (size\_t i = 0; i < left.rows; i++) {  
 temp.clear();  
 **auto** lvec = left.data.at(i);  
 **auto** rvec = right.data.at(i);  
 temp.resize(lvec.size());  
 std::transform(lvec.begin(), lvec.end(), rvec.begin(), temp.begin(),std::plus<T>());  
 result.data.push\_back(temp);  
 }  
  
 **return** result;  
}  
  
**template**<**typename** T>  
Matrix<T> **operator**\*(**const** Matrix<T>& left, **const** std::vector<T>& right) {

if (right.rows != left.cols){

throw std::range\_error("matrix rows/cols is out of range");

}

Matrix<T> result(1, left.rows);  
 std::vector<T> temp;  
  
 **for** (size\_t i = 0; i < left.rows; i++) {  
 **auto** vec = left.data.at(i);  
 **int** sum = 0;  
  
 **for** (size\_t index = 0; index < left.cols; index++) {  
 sum += vec.at(index) \* right.at(index);  
 }  
 temp.clear();  
 temp.push\_back(sum); *// to make it vector with one element* result.data.push\_back(temp); *// to make it our vectors of vectors* }  
 **return** result;  
}

# **שאלה 7 [22 נק']**

כיתבו מחלקה בשם Money, המייצגת סכום-כסף מסוים במטבע מסוים. המחלקה צריכה לאפשר חיבור של סכומי-כסף במטבעות שונים, תוך המרה אוטומטית של יחידות.

שער-החליפין של כל מטבע, בשקלים, נקבע בתחילת התוכנית.

מצורפת תוכנית דוגמה. עליכם לממש את כל הפונקציות הדרושות כדי שהתוכנית תעבוד בהתאם להערות המופיעות בגוף הקוד. אין צורך לממש פונקציות נוספות. הערות:

* אין צורך להתייחס לשגיאות עיגול של מספרים ממשיים בפלט.
* עליכם לכתוב שני קבצים: Money.hpp ו- Money.cpp .

#include <iostream>

using namespace std;

#include "Money.hpp"

using ariel::Money;

int main() {

// Set the exchange rates in shekels:

Money::set\_rate("ILS", 1);

Money::set\_rate("USD", 4);

Money::set\_rate("EUR", 6);

// Define amounts and do calculations:

Money shekel1(1, "ILS");

Money dollar3(3, "USD");

Money sum = shekel1 + dollar3;

cout << sum << endl; // 13 ILS

cout << (dollar3 + Money(1, "EUR")) << endl; // 4.5 USD

(sum += Money(2, "EUR")) += Money(0.5, "ILS");

cout << sum << endl; // 25.5 ILS

// Money(1, "XXX"); // throws an exception

}

פתרון אפשרי בעמוד הבא. **מפתח ניקוד**:

* סעיף א [11 נק']. קובץ הכותרת;
* סעיף ב [11 נק']. קובץ המימוש.

מלוא הנקודות ניתנו רק לפתרון קצר יעיל וכללי, למשל פתרון ששמר את שערי-החליפין ב-map באופן המאפשר להוסיף מטבעות נוספים (מאפשר קריאה וכתיבה בשורה אחת), במקום לכתוב פקודת תנאי ארוכה עם 3\*3 מקרים.

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*\* Money.hpp \*\*\*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/***#include** <string>  
**#include** <map>  
**#include** <iostream>  
**using** std::string, std::map, std::endl, std::ostream, std::ifstream;  
  
**namespace** ariel{  
  
  
**class** Money {  
 **double** amount;  
 string currency;  
 **double** rate;  
  
 **static** map<string,**double**> map\_currency\_to\_rate;  
  
**public**:  
 Money(**double** amount, string currency): amount(amount), currency(currency) {   
 rate = map\_currency\_to\_rate.at(currency); // throws exception if not found

}  
  
 **friend** Money **operator**+(const Money& a, const Money& b);  
 **friend** Money& **operator**+=(Money& a, const Money& b);  
 **friend** ostream& **operator**<< (ostream& out, const Money& m);  
  
 **static void** set\_rate(string currency, **double** rate) {  
 map\_currency\_to\_rate[currency] = rate;  
 }  
};  
}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*\* Money.cpp \*\*\*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/***#include "Money.hpp"  
  
namespace** ariel{  
  
  
Money **operator**+(const Money& a, const Money& b) {  
 **double** amount = a.amount + b.amount\*b.rate/a.rate;  
 **return** Money(amount, a.currency);  
}  
  
Money& **operator**+=(Money& a, const Money& b) {  
 a.amount += b.amount\*b.rate/a.rate;  
 **return** a;  
}  
  
ostream& **operator**<< (ostream& out, Money m) {  
 **return** (out << m.amount << **" "** << m.currency) << endl;  
}  
  
map<string,**double**> Money::map\_currency\_to\_rate{};  
  
  
}

# **שאלה 8 - מענקים**

א. האם השתמשנו בבדיקות שלכם באחת המטלות? אם כן אנא ציינו באיזו מטלה בדיוק.

**[3 נקודות]**

ב. האם זכיתם באחד משלושת המקומות הראשונים בתחרות בול-פגיעה? אם כן ציינו באיזה מקום.

**[4 נקודות למקום ראשון, 3 נקודות למקום שני, 2 נקודות למקום שלישי]**

ג. האם הודענו לכם שנשתמש בפתרון שלכם למטלה 6 בשנה הבאה?

**[5 נקודות]**