פקולטה: מדעי הטבע
מחלקה: מדעי המחשב
שם הקורס: מבנה זיכרון ושפת C++
קוד הקורס: 7027810-2 כל הקבוצות - קבוצות 1,3,4,5
מועד __ב___ סמסטר ___ב___תאריך בחינה:____26/7/2018_
משך הבחינה: 3 שעות
שם המרצים: אראל סגל-הלוי, חרות סטרמן

יש לענות על כל השאלות במחברת הבחינה.
אין צורך להעתיק את השאלון למחברת - השאלון יתפרסם במודל לאחר הבחינה.

בשאלות 1-5, כששואלים "מה סוג השגיאה", יש לכתוב אחת מהאפשרויות הבאות:

- שגיאת קומפילציה (compilation error)
- שגיאת קישור (link error)
- שגיאת זמן-ריצה (runtime error)
- שגיאה לוגית (logic error)

כששואלים "מה גורם לשגיאה" יש לפרט ולציין מספרי שורות בהתאם לקוד המובא בשאלה.

כששואלים "איך לתקן את השגיאה" יש לציין מספרי שורות ולהסביר מה צריך לשנות בכל שורה/שורות. בחלק מהמקרים ישנם כמה תיקונים אפשריים. ניקוד מלא יינתן רק לתיקון הטוב ביותר מנקודת מבט של הנדסת תוכנה, כפי שנלמד בכיתה.

- ***אין לשנות את התוכנית הראשית.***

יש לענות תשובות מלאות אך ממוקדות. לא יינתנו נקודות על תשובות עם טקסט מיותר שאינו קשור לנושא.

ניתן לענות על כל שאלה במילים "לא יודע" או "לא יודעת". במקרה זה השאלה לא תיבדק כלל, ואתם תקבלו 20% מהניקוד על אותה שאלה (מעוגל כלפי מטה).

אסור להשתמש בכל חומר עזר.

בהצלחה!!

שאלה1 [10 נק']

נתונה התוכנית:

```
01   #include <string>
02   #include <iostream>
03   using namespace std;

04   class WaterCreature {
05     string name;
06   public:
07     WaterCreature(string name): name(name) {}
08     string getName() { return name; }
09     void swim() { cout << "I swim! "; }
10   };

11   class EarthCreature {
12     string name;
13   public:
14     EarthCreature(string name): name(name) {}
15     string getName() { return name; }
16     void run() { cout << "I run! "; }
17   };

18   class Frog: public WaterCreature, public EarthCreature {
19     Frog(string name):WaterCreature(name),EarthCreature(name) {}
20   };

21   int main() {
22     Frog f("froggy");
23     return 0;
24   }
```

כשמריצים make, מתקבל הפלט הבא:

```
main.cpp:22:7: error: calling a private constructor of class 'Frog'
Frog f("froggy");
^
```

א [5 נק']. מה סוג השגיאה (קומפילציה / קישור / זמן-ריצה / לוגית)? מה בדיוק גורם לשגיאה?

- *שגיאת קומפילציה. הבנאי של `Frog` הוא פרטי כי לא הוגדר כציבורי (במחלקה כל שיטה היא פרטית אלא אם כן היא מוגדרת כציבורית. הבנאים במחלקות המורישות הם ציבוריים, אבל בנאים לא עוברים בירושה).*

ב [5 נק']. איך אפשר לתקן את הבאג ע"י שינוי/הוספה של שורה אחת? (לא בתוכנית הראשית).

- *להוסיף `public`: בין שורה 18 לשורה 19.*

שאלה 2 [10 נק']

נתון הקוד החלקי הבא (חלק מהשורות הושמטו):

```
01   template<typename T>
02   class Matrix {
03     T* values;
04     int rows,cols;
05   public:
06     Matrix(int rows, int cols):
07       rows(rows), cols(cols),
08       values(new T[rows*cols])
09       {}
10   };
11
12   int main() {
13     for (int i=0; i<10000; ++i) {
14       Matrix<int> m1 (1000,1000);
15       // do some calculations with m1
16     }
17       return 0;
18   }
```

כשמריצים את התוכנית, לאחר מספר שניות המחשב מתחיל לעבוד יותר לאט, המאוורר שלו משמיע קולות של מאמץ, ובסופו של דבר מקבלים הודעת שגיאה: `Out of Memory`.

א [5 נק']. מה בדיוק גורם לשגיאה? פרטו.

- *דליפת זיכרון [1 נק']. במחלקה `Matrix` הבנאי מקצה מקום בזיכרון ע"י `new`, אבל לא מוגדר מפרק, ולכן המפרק שלה הוא ברירת המחדל - לא עושה כלום [2 נק']. בלולאה בתוכנית הראשית, יוצרים 10000 מטריצות ומפרקים אותן, מקצים הרבה מקום אבל לא משחררים אותו, לכן בסופו של דבר הזיכרון נגמר [2 נק'].*

ב [5 נק']. איך אפשר לתקן את הבאג בלי לשנות את התוכנית הראשית? פרטו.

- *להוסיף מפרק שמוחק את `values` בין שורה 09 לשורה 10:*

```
~Matrix(){
      delete[] values;
  }
```

שגיאות אפשריות:

- לשכוח סוגריים במחיקה [1 נק'].
- למחוק בלולאה [2 נק'].

שאלה 3 [10 נק']

נתונה התוכנית הבאה:

```
01   class XX {
02   public:
03     void aa() {}
04     virtual void bb() {}
05     virtual void cc() {}
06     virtual void dd()=0;
07     static void ee() {}
08   };
09   class YY: public XX {
10   public:
11     void aa() {}
12     virtual void bb() {}
13     void cc() {}
14     void dd() {}
15     virtual void ee() {}
16   };
17   int main() {
18     XX* zz = new YY();
19     zz->aa();
20     zz->bb();
21     zz->cc();
22     zz->dd();
23     zz->ee();
24     return 0;
25   }
```

ידוע שהתוכנית מתקמפלת בלי שגיאות.

בסעיפים הבאים, יש לבחור את הפונקציות המתאימות מתוך הרשימה:

```
XX::aa, XX::bb, XX::cc, XX::dd, XX::ee
YY::aa, YY::bb, YY::cc, YY::dd, YY::ee
```

א [2 נק']. איזה פונקציות מהרשימה הנ"ל נמצאות בטבלת הפונקציות הוירטואליות של המחלקה XX?

- ***פתרון***: *`XX::bb, XX::cc`. (הפונקציות `aa, ee` לא וירטואליות, והפונקציה `dd` לא ממומשת).*
  - *מי שבכל-זאת כתב את `XX::dd` קיבל נקודה אחת.*

ב [2 נק']. איזה פונקציות מהרשימה הנ"ל נמצאות בטבלת הפונקציות הוירטואליות של המחלקה YY?

- ***פתרון***: *`YY::bb, YY::cc, YY::dd, YY::ee` (כולן וירטואליות חוץ מ `aa`).*

ג [2 נק']. איזה פונקציות מהרשימה הנ"ל נקראות מהתוכנית הראשית בזמן ריצתה?

- ***פתרון***: *`XX::aa, YY::bb, YY::cc, YY::dd, XX::ee` - לפי כללי קריאה לפונקציות וירטואליות.*

כדי לפתור את סעיפים ד, ה, היה צריך לדעת איך פועל המנגנון של קריאה לפונקציות רגילות ווירטואליות (למדנו בשבוע 6 כולל הדגמות באסמבלי). מי שעשה את מטלה 2 יכל גם לזכור שהפקודה `callq` משמעה קריאה לפונקציה, אבל היה אפשר גם להבין את זה מתוך ההקשר.

ד [2 נק']. מהו קוד האסמבלי המתאים לשורה 19 בתוכנית הראשית? בחרו את התשובה ההגיונית ביותר.

- ***פתרון***: *בשורה 19 קוראים לפונקציה `aa` שהיא לא וירטואלית. במצב זה הקומפיילר מכניס קריאה ישירה לפונקציה בהתאם לסוג המשתנה. במקרה זה סוג המשתנה הוא XX ולכן הקומפיילר מכניס קריאה לפונקציה `XX::aa`. האפשרות היחידה שמתאימה כאן היא האפשרות הראשונה - רק שם נזכרת באופן ישיר הפונקציה `XX::aa`.*

<u>Assembly code A:</u>

```
movq -8(%rbp), %rdi
callq XX::aa()
```

ה [2 נק']. מהו קוד האסמבלי המתאים לשורה 20 בתוכנית הראשית? בחרו את התשובה ההגיונית ביותר.

- ***פתרון***: *בשורה 20 קוראים לפונקציה `bb` שהיא וירטואלית. במצב זה הקומפיילר לא מכניס קריאה לפונקציה מסויימת, אלא מכניס קוד שניגש למצביע `vptr` ומשם לטבלת הפונקציות הוירטואליות ובוחר את הכניסה המתאימה. האפשרות היחידה שמתאימה כאן היא האפשרות השלישית - רק שם יש קוד מורכב שקורא לפונקציה באופן עקיף בלי להזכיר פונקציה ספציפית:*

<u>Assembly code C:</u>

```
movq -8(%rbp), %rax
movq (%rax), %rdx
movq %rax, %rdi
callq *(%rdx)
```

שאלה 4 [10 נק']

נתונה תוכנית המורכבת משלושה קבצים.

XX.h:

```
01   #include <iostream>
02
03   class XX {
04     int myval;
05   public:
06     XX(int newval): myval(newval) {}
07     int val() const;
08     friend std::ostream& operator<< (std::ostream&, const XX&);
09   };
10
11   std::ostream& operator<< (std::ostream& out, const XX& xx) {
12     out << xx.myval;
13     return out;
14   }
```

XX.cpp:

```
21   #include "XX.h"
22   int XX::val() const {
23     return myval;
24   }
```

main.cpp:

```
31   #include "XX.h"
32
33   int main() {
34     XX oxx(5);
35     std::cout << oxx << std::endl; // should print 5;
36       return 0;
37   }
```

כשמריצים את הפקודה:

```
clang++-5.0 -std=c++17 *.cpp
```

מתקבלת הודעת השגיאה הבאה:

```
/tmp/main-f7c5da.o: In function `operator<<(std::ostream&, XX
const&)':
main.cpp:(.text+0x0): multiple definition of
`operator<<(std::ostream&, XX const&)'
/tmp/XX-25846f.o:XX.cpp:(.text+0x0): first defined here
```

א [5 נק']. מה סוג השגיאה (קומפילציה / קישור / זמן-ריצה / לוגית)? מה בדיוק גורם לשגיאה?

- *שגיאת קישור [2 נק'].*
- *אופרטור הפלט מוגדר בקובץ הכותרת (שורות 11-14). הקובץ הזה מוכלל בשתי יחידות קומפילציה שונות - ב `main.cpp` וב `XX.cpp`. כל קובץ מתקמפל בסדר, אבל כשהמקשר בא לקשר אותם, הוא רואה שיש כפילות - האופרטור מוגדר פעמיים - ולכן צועק על `multiple definition`.*

ב [5 נק']. איך אפשר לתקן את הבאג? פרטו.

- *להעביר את המימוש של אופרטור הפלט (שורות 11-14), מהקובץ `XX.h` לקובץ `XX.cpp`.*
- *בקובץ הכותרת להשאיר רק הצהרה בלי מימוש:*

```
std::ostream& operator<< (std::ostream& out, const XX& xx);
```

- *הפתרון הזה הוא גם הפתרון הנכון מנקודת מבט של הנדסת תוכנה - הפרדת קבצים בין הצהרה לבין מימוש - כפי שלמדנו בכיתה.*
- *פתרון לא נכון: להוסיף `pragma once`. ההוראה הזאת מתייחסת רק לקומפיילר ולא משפיעה על שלב הקישור - נסו ותראו.*

שאלה 5 [10 נק']

נתונה התוכנית:

```
01   #include <vector>
02   #include <iostream>
03   using namespace std;
04
05   template<typename T>
06   class TwoVectors {
07     vector<T> theVectors[2];
08   public:
09     int sizeOfVector(int i) {
10       return theVectors[i].size();
11     }
12   };
13
14   template<typename T>
15   ostream& operator<< (ostream& out, const TwoVectors<T>& v) {
16     out << "first vector has size " << v.sizeOfVector(0) << endl;
17     out << "2nd vector has size " << v.sizeOfVector(1) << endl;
18     return out;
19   }
20
21   int main() {
22     TwoVectors<int> v;
23     cout << v << endl;
24       return 0;
25   }
```

כשמריצים make מתקבלת ההודעה הבאה:

```
main.cpp:16:37: error: member function 'sizeOfVector' not viable:
'this' argument has type 'const TwoVectors<int>', but function is
not marked const
out << "first vector has size " << v.sizeOfVector(0) << endl;
^
main.cpp:23:7: note: in instantiation of function template
specialization 'operator<<<int>' requested here
cout << v << endl;
^
```

א [5 נק']. מה סוג השגיאה (קומפילציה / קישור / זמן-ריצה / לוגית)? מה בדיוק גורם לשגיאה?

- *שגיאת קומפילציה.*
- *הפונקציה `sizeOfVector` בשורה 9 לא מוגדרת כקבועה.*
- *אבל `v` כן מוגדר כקבוע בשורה 15 .*
- *בשורה 16 מנסים להפעיל פונקציה לא קבועה על עצם קבוע והקומפיילר לא מאפשר את זה.*

ב [5 נק']. מהי הדרך הנכונה ביותר לתקן את הבאג?

- *כיוון שהפונקציה `sizeOfVector` לא משנה את העצם, הכי פשוט להגדיר אותה כ-`const` בשורה 9.*

```
09     int sizeOfVector(int i) const {
```

- *פתרון פחות טוב הוא להוריד את ה-`const` בשורה 15 - זה לא נכון לפי כללי הנדסת תוכנה. על פתרון זה ניתנו 3 נק'.*

שאלה 6 [10 נק']

כיתבו פונקציה בשם `integral` המקבלת שני פרמטרים: פונקציה f ומספר ממשי x, ומחשבת את האינטגרל של הפונקציה f בין אפס לבין x, כסכום של שטחי מלבנים ברוחב 0.001.
למשל, אם מוגדרת הפונקציה:

```
double twice(double x) { return 2*x; }
```

אז הקריאה:

```
integral(twice, 10)
```

תחזיר את הסכום:

```
0.001 * (2*0 + 2*0.001 + 2*0.002 + ... + 2*9.998 + 2*9.999 + 2*10)
```

(הסכום הזה בערך שווה 100 - האינטגרל של 2x בין 0 ל-10).

לפניכם תוכנית ראשית לדוגמה. עליכם לכתוב את התוכן של הקובץ **`integral.h`** בלבד כך שהתוכנית הראשית תתקמפל ותיתן את התוצאות הנכונות. אין צורך לכתוב את הקובץ `integral.cpp`.

```
#include "integral.h"
#include <iostream>
using namespace std;

double twice(double x) { return 2*x; }

struct fourtimes {
  double operator() (double x) {return 4*x;}
};

int main() {
  cout << integral(twice, 10) << endl; // ~100 (x^2 from 0 to 10)
  cout << integral(fourtimes{}, 10) << endl; // ~200 (2*x^2 0 to 10)
  cout << integral([](double x){return 6*x;}, 10) << endl; // ~300

  cout << integral(twice, 20) << endl; // ~400 (x^2 from 0 to 20)
  cout << integral(fourtimes{}, 20) << endl; // ~800 (2*x^2 0 to 20)
  cout << integral([](double x){return 6*x;}, 20) << endl; // ~1200
    return 0;
}
```

פתרון: *הרעיון מאד דומה למה שעשינו בשיעור על template כשמימשנו נגזרת של פונקציה כללית - שבוע 9 בגיטהאב. צריך להגדיר תבנית-פונקציה שיכולה לקבל כל פרמטר, בתנאי שיש לו אופרטור סוגריים. כך הפונקציה תומכת גם בפונקציות פרימיטיביות, גם במחלקות עם אופרטור סוגריים, וגם בביטויי למדא.*

```
template<typename Function>

double integral(Function f, double xx) {
   double sum=0;
   for (double x=0; x<=xx; x+=0.001) {
      double y = f(x);
      sum += y*0.001;
   }
   return sum;
}
```

מפתח ניקוד:

- *שימוש נכון ב template [2 נק'].*
- *חתימת פונקציה נכונה [2 נק'].*
- *לולאה נכונה [2 נק'].*
- *חישוב הסכום נכון [2 נק'].*
- *ערך מוחזר נכון [2 נק'].*

הערה: *על פונקציה בלי template, שעובדת נכון על רק פונקציות פרימיטיביות, ניתנו 6 נק'.*

שאלה 7 [40 נק']

עליכם לכתוב מחלקה בשם RunningAverage, המאפשרת להוסיף לתוכה מספרים ולחשב את הממוצע של כל המספרים שהוכנסו לתוכה עד כה. **סיבוכיות הזיכרון הדרושה היא O(1)**.

נתונה תוכנית ראשית. עליכם לכתוב את הקבצים `RunningAverage.h` ו `RunningAverage.cpp` כך שהתוכנית הראשית תעבוד. אין צורך לממש אופרטורים שאינם נזכרים בתוכנית הראשית. הקפידו על חלוקה נכונה של הקוד בין הקבצים, תיעוד, הסתרת מידע, וכל שאר כללי הנדסת תוכנה שנלמדו בקורס.

```
#include "RunningAverage.h"
#include <iostream>
using namespace std;

int main() {
  RunningAverage a;
  a += 8;
  a += 7;
  cout << a << endl; // 7.5 (average of 7 and 8)
  (a += 6.5) += 6.5;
  cout << a << endl; // 7 (average of 7, 8, 6.5 and 6.5)
    a = a + 9 + 9 + 9 + 9; // 8 (average of the above and 9,9,9,9)
    return 0;
}
```

פתרון: *כיוון שדרושה סיבוכיות O(1), לא צריך לשמור את כל המספרים, אלא רק את הסכום המצטבר ואת כמות המספרים המצטברת. צריך לאתחל אותם בבנאי ולעדכן אותם באופרטורים:*

RunningAverage.h:

```
#include <iostream>


class RunningAverage {
  double sum;
  int count;
public:
  RunningAverage(): sum(0.0), count(0)  {  }
  RunningAverage& operator+= (double d);
  const RunningAverage operator+ (double d) const;
  double avg() const;
};

std::ostream& operator<< (std::ostream& out, const RunningAverage& r);
```

RunningAverage.cpp:

```
#include "RunningAverage.h"

RunningAverage& RunningAverage::operator+= (double d) {
  sum += d;
  count++;
  return *this;
}
const RunningAverage RunningAverage::operator+ (double d) const {
  RunningAverage result;
  result.sum = sum;
  result.count = count;
  result += d;
  return result;
}
double RunningAverage::avg() const {
  return sum/count;
}
std::ostream& operator<< (std::ostream& out, const RunningAverage& r) {
  return (out << r.avg());
}
```

מפתח ניקוד:

שימו לב - לנוחותכם הניקוד על השאלה חולק ל-5 חלקים, כל חלק 8 נקודות.

- 7.1 משתני המחלקה [8 נק'].
  - סיבוכיות זיכרון נכונה [5 נק'].
  - הגדרה נכונה של `public/private` [3 נק'].
    - על משתנים סטטיים הורדו 2 נק'.
- 7.2 הצהרות נוספות בקובץ h [8 נק'].
  - בנאי נכון [5 נק'].
  - הגדרה נכונה של `const` [1 נק' לכל אחד, סה"כ 3].
    - הורדה 1 נק' על שימוש לא נכון ב-`friend`.
- 7.3 מימוש אופרטור += נכון [8 נק'].
  - יש לשנות את האובייקט הנוכחי [6 נק'].
  - החזרת `this` לצורך שירשור [2 נק'].
- 7.4 מימוש אופרטור + נכון [8 נק'].
  - יש להשאיר את האובייקט הנוכחי ללא שינוי [6 נק'].
  - החזרת הערך הנכון [2 נק'].
- 7.5 מימוש אופרטור >> נכון [8 נק'].
  - ערך מוחזר [2 נק'].