שפת C++ – תרגיל 1

היכרות עם השפה, references , const , iostream ,classes,

dynamic allocation, operators overloading

תאריך הגשה: יום רביעי 26.8.15 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): יום חמישי 27.8.15 עד שעה 23:55

תאריך ההגשה של הבוחן: יום רביעי 26.8.15 עד שעה 23:55

1. הנחיות חשובות:

1. בכל התרגילים יש לעמוד בהנחיות הגשת התרגילים וסגנון כתיבת הקוד. שני המסמכים נמצאים באתר הקורס – הניקוד יכלול גם עמידה בדרישות אלו.
2. בכל התרגילים עליכם לכתוב קוד ברור. בכל מקרה בו הקוד שלכם אינו ברור מספיק עליכם להוסיף הערות הסבר בגוף הקוד. יש להקפיד על תיעוד (documentation) הקוד ובפרט תיעוד של כל פונקציה.
3. במידה ואתם משתמשים בעיצוב מיוחד או משהו לא שגרתי, עליכם להוסיף הערות בקוד המסבירות את העיצוב שלכם ומדוע בחרתם בו.
4. בכל התרגילים במידה ויש לכם הארכה ואתם משתמשים בה, **חל איסור להגיש קובץ כלשהוא בלינק הרגיל (גם אם לינק ההגשה באיחור טרם נפתח)**. מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.
5. אין להגיש קבצים נוספים על אלו שתדרשו.
   a. אין להגיש קובץ README אלא אם צוין במפורש שיש צורך בכך (לדוגמא, בתרגיל זה אין צורך להגיש).
   b. עליכם לקמפל עם הדגלים Wall -Wextra -Wall ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה בציון התרגיל.** למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex1.c יש להריץ את הפקודה:

```
g++ -std=c++11 -Wextra -Wall ex1.cpp -o ex1
```

6. עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי bit-64 (מחשבי האקוריום, לוי, השרת river). **חובה להריץ את התרגיל במחשבי בית הספר לפני ההגשה**. (ניתן לוודא שהמחשב עליו אתם עובדים הנו בתצורת bit-64 באמצעות הפקודה "uname -a" ווידוא כי הארכיטקטורה היא 64, למשל אם כתוב x86_64)
7. לאחר ההגשה, בדקו הפלט המתקבל בקובץ ה-PDF שנוצר מה presubmission script בזמן ההגשה. באם ישנן שגיאות, תקנו אותן על מנת שלא לאבד נקודות.
   שימו לב ! תרגיל שלא יעבור את ה presubmission script ציונו ירד משמעותית (הציון יתחיל מ-50, ויוכל לרדת) **ולא יהיה ניתן לערער על כך.**
8. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. חישבו על מקרי קצה לבדיקת הקוד.

9. **הגשה מתוקנת -** לאחר מועד הגשת התרגיל ירוצו הבדיקות האוטמטיות ותקבלו פירוט על הטסטים בהם נפלתם. לשם שיפור הציון יהיה ניתן להגיש שוב את התרגיל לאחר תיקוני קוד קלים ולקבל בחזרה חלק מהנקודות - פרטים מלאים יפורסמו בפורום ואתר הקורס.

2. הנחיות חשובות לכלל התרגילים בקורס ++C

1. הקפידו להשתמש בפונקציות ואובייקטים של ++C (למשל new, delete, cout) על פני פונקציות של C (למשל malloc, free, printf). בפרט השתמשו במחלקה string (ב-std::string) ולא במחרוזת של (char *) C.
2. יש להשתמש בספריות סטדנרטיות של ++C ולא של C אלא אם כן הדבר הכרחי (וגם אז עליכם להוסיף הערה המסבירה את הסיבות לכך).
3. הקפידו על עקרונות Information Hiding – לדוגמא, הקפידו כי משתני המחלקות שלכם מוגדרים כמשתנים פרטיים (private).
4. הקפידו לא להעתיק by value משתנים כבדים, אלא להעבירם (היכן שניתן) by reference.
5. **<u>הקפידו מאוד</u>** על שימוש במילה השמורה const בהגדרות הפונקציות והפרמטרים שהן מקבלות.
   פונקציות שאינן משנות פרמטר מסויים – הוסיפו const לפני הגדרת הפרמטר.
   מתודות של מחלקה שאינן משנות את משתני המחלקה – הוסיפו const להגדרת המתודה.
   שימו לב: הגדרת משתנים / מחלקות ב- ++C כקבועים הוא אחד העקרונות החשובים בשפה.
6. הקפידו על השימוש ב- static, במקומות המתאימים (הן במשתנים והן במתודות)
7. הקפידו לשחרר את כל הזיכרון שאתם מקצים (**השתמשו ב-valgrind כדי לבדוק שאין לכם דליפות זיכרון**).

3. מידע חשוב נוסף:

1. ניתן להתחבר באמצעות SSH למחשבי בית הספר (למשל לשם בדיקת הקוד לפני הגשה מהבית)
   http://wiki.cs.huji.ac.il/wiki/Connecting_from_outside
2. עליכם להכיר את ספריית הקלט-פלט של שפת C ובייחוד את השימוש בפונקציות scanf ו printf
   http://www.cplusplus.com/reference/clibrary/cstdio

4. IntMatrix :

1. <u>רקע</u>

עליכם לכתוב את הקבצים IntMatrix.h, IntMatrix.cpp ו IntMatrixDriver.cpp שיחוברו יחד לכדי תכנית IntMatrixMainDriver שתבצע פעולות חישוב פשוטות על מטריצות שערכי התאים שלהן הם מספרים שלמים.

2. <u>יישום הממשק IntMatrix</u>

a) בקובץ IntMatrix.h עליכם להגדיר את המחלקה IntMatrix, שתתאר מטריצה שאיבריה מספרים שלמים (מטיפוס int). המימושים לפונקציות המחלקה יהיו בקובץ.IntMatrix.cpp.

b) עליכם להצהיר ולממש במחלקה את הפונקציות הבאות:

- בנאי ברירת מחדל (ללא ארגומנטים).
- בנאי העתקה.
- בנאים נוספים כרצונכם.
- destructor.
- אופרטור השמה ('=') לשם ביצוע פעולת השמת מטריצה. אופרטור זה מאפשר שינוי של המטריצה המיוצגת על ידי האובייקט שמשמאל לסימן ה-'=', כך שתהייה זהה למטריצה המיוצגת על ידי האובייקט המועבר כפרמטר (מימין לסימן ה-'='). זכרו: פעולת ההשמה גורמת להיווצרות עותק זהה ובלתי תלוי.
- אופרטורי חיבור ('+' ו- '=+') - לשם ביצוע פעולות חיבור מטריצות.
- אופרטורי חיסור ('-' ו- '=-') לשם ביצוע פעולות חיסור מטריצות.
- אופרטורי כפל ('*' ו- '=*') לשם ביצוע פעולות כפל מטריצות.
- פונקצית שחלוף בשם trans. הפונקציה אינה משנה את האובייקט עליו היא הופעלה, אלא מחזירה אובייקט חדש.
- פונקצית עקבה בשם trace המחזירה ערך סקלארי – מספר שלם.
- מימוש אופרטור '<<' לשם הדפסת המטריצה עם אובייקט ostream (ראו אופן הדפסת המטריצה בהמשך).
- עוד פונקציות שנראה לכם שכדאי שיוגדרו במחלקה (הפונקציות הללו <u>לא</u> <u>מספיקות</u> כדי לבצע את כל מה שהתכנית צריכה. יהיה עליכם להחליט איך להגדיר את הממשק במלואו, אלו פונקציות נוספות המחלקה תבצע בעצמה, ואלו פונקציות הדרייבר אמור לעשות מחוץ למחלקה).

c) טיפים והנחיות:

- כל הפונקציות המתאימות מבצעות את הפעולה המתמטית המקבילה להגדרתם.
- בפונקציית trace אתם יכולים להניח שהמטריצה ריבועית (באחריות מי שקורא למתודה לוודא זאת).
- בכל האופרטורים אתם יכולים להניח שממדי המטריצות מתאימים (באחריות מי שמשתמש באופרטור לוודא שהמטריצות ממימדים מתאימים).

- בכל האופרטורים עם הסימן = מתבצע עדכון האובייקט השמאלי. חישבו היטב מה קורה מבחינת הזכרון כאשר המטריצה משנה את מימדיה עקב פעולה זו.
- הממשק המדויק (החתימות של הפונקציות) לא מוכתב לכם ונתון להחלטתכם, אבל ברוב המקרים ישנה דרך עיקרית אחת שהיא הטובה ביותר להגדרת הפונקציה. חישבו למשל על:
  - האם על הפונקציה להיות מוגדרת כ const.
  - האם הארגומנט צריך להיות מועבר by reference או by value, או אולי כדאי להשתמש במצביע. האם האגרומנט צריך להיות מוגדר כ const?
  - ערך ההחזרה, האם הוא צריך להיות מוגדר כ const, והאם הוא מועבר by reference או by value.
  - חישבו איך האופרטור שאתם מממשים פועל על טיפוסים מובנים בשפה ונסו להתחקות אחרי זה במימוש שלכם עבור המטריצה.
- השתמשו ב- asserts לשם ווידוא הנחות היסוד שלכם (למשל:בתוך מתודות המחלקה, אנחנו מניחים שהמשתמש הפעיל את המתודה/אופרטור על מטריצות במימדים מתאימים, אך ניתן לוודא זאת עם assert ויש לכך עדיפות על פני גישה לזכרון שאינו באמת שייך לאובייקט).

3. <u>יישום דרייבר IntMatrixDriver</u>

a) בקובץ IntMatrixDriver.cpp עליכם לממש את הדרייבר של התכנית.

b) עם הרצת התכנית יוצג תפריט למשתמש, כדי לבחור פעולה אחת לביצוע על מטריצות:

Choose operation:
1. add
2. sub
3. mul
4. trans
5. trace

c) התכנית תצפה לקבל קלט מהמשתמש – מספר שלם 1 עד 5 (כמובן על התכנית לוודא שהמספר שהתקבל הוא בין 1-5) דרך הטיפול/תגובה היא לבחירתכם (אבל עליכם להמשיך לנסות לקבל קלט מהמשתמש עד אשר תקבלו קלט תקין). אתם רשאים להניח שהקלט יהיה מספר שלם.

d) אם נבחרה פעולה 1-3 התכנית תציג את ההודעה:

```
Operation [add|mul|sub] requires 2 operand matrices.
```

ואזי המשתמש יתבקש להזין לתכנית שתי מטריצות (שכן פעולות אלו הן בינאריות – יש להן שני אופרנדים).

e) אם נבחרה אופציה 4-5 יתבקש המשתמש להזין רק מטריצה אחת (שכן פעולות אלו מבוצעות על אופרנד אחד).

```
Operation [transpose|trace] requires 1 operand matrix
```

f) לכל מטריצה המשתמש מתבקש להזין מספר שורות, מספר עמודות, ואז להזין שורה אחרי שורה את ערכי השורה מהמטריצה, כשלאחר כל איבר בשורה יש פסיק. **שימו לב! אינכם יכולים להניח שמספר השורות ומספר העמודות חסום בערך מסויים.**

לדוגמא: להלן הזנה של 2 מטריצות:

```
3
2
10,100,
3,2,
12,1,
2
3
14,-2,1,
0,30,-10,
```

g) השתמשו בפתרון בית הספר כדי להעתיק במדויק את ההודעות שיש להציג למשתמש וכדי להבין את הפורמט שבו המשתמש צריך להזין את ערכי המטריצות במידה ועדיין יש לכם ספק. (לכל מטריצה המשתמש מתבקש להזין מספר שורות, מספר עמודות, ואז להזין שורה אחרי שורה את ערכי השורה מהמטריצה, כשלאחר כל איבר בשורה יש פסיק).

h) על הדרייבר לייצר אובייקטים של מטריצות מהמחלקה שמימשתם (אובייקט אחד או שניים, בהתאם לאופציה שנבחרה), לבצע את החישוב, להציג את המטריצות שנקלטו ע"י התכנית ולהציג את תוצאת חישוב הפעולה שנבחרה.

i) הדפסת מטריצה תהיה שורה אחר שורה, ובין כל שני איברים בשורה יהיה רווח יחיד.

למשל להלן הדפסת תוצאת חיבור 2 מטריצות:

```
==========
Resulted matrix:

20 912 -6
-46 22 14
```

j) אם תוצאת החישוב היא מטריצה (עבור פעולות החיבור, חיסור, כפל או שחלוף) תוצג הודעה מתאימה (הפעילו את פתרון בית הספר או ראו קבצי פלט לדוגמה כדי לראות בפירוט) ותודפס מטריצת התוצאה.

k) אם תוצאת החישוב היא מספר שלם תוצג הודעה מתאימה ויוצג ערך התוצאה.

l) לבסוף התכנית מסיימת את ריצתה ומחזירה ערך 0.

4. <u>הנחיות נוספות:</u>

- שימו לב למקרים בהם הקלט לבחירת הפעולה לא תקין (כלומר לא ניתן לבצע את הפעולה שנבחרה קודם לכן) וודאו כי הפלט של תוכניתכם במקרה כזה זהה לפלט של פתרון בית הספר. ספציפית:
  - אם מימדי שתי המטריצות שמבקשים לחבר/לחסר/להכפיל לא מתאימים עליכם לאתר זאת ולהדפיס את ההודעת שגיאה הבאה:

```
Error: [add|mul|sub] failed - different dimensions.
```

- אם מימד ההמטריצה שמבקשים לחשב את העקבה שלה אינה ריבועית עליכם להדפיס הודעת שגיאה:

```
Error: trace failed - The matrix isn't square.
```

- אתם יכולים להניח כי בכל שלב בו מוזנת מטריצה ע"י המשתמש, ערכי המטריצה יוזנו בצורה תקינה (כלומר אינכם צריכים להתמודד עם מקרים בהם ערכי המטריצה לא יתאמו במספר השורות/עמודות למה שהוזן, או עם מצב בו לא יוזנו מספר השורות/עמודות).
- להזכירכם, ניתן להזין קלט מקובץ באמצעות הפניית ה- standard input מה-shell. למשל:

```
IntMainDriver < 'input file name'
```

- **שימו לב! : בתרגיל זה אין להשתמש במבנה נתונים כלשהו מספרית STL** (למשל vector) **או מכל קוד חיצוני אחר.**

5. חומר עזר:

1. את פתרון הבית ספר ניתן למצוא ב:

```
~slabcpp/www/ex1/schoolSol.tar
```

הריצו אותו כדי לראות איך התכנית צריכה להתנהג

2. דוגמאות לטסטים ניתן למצוא ב:

```
~slabcpp/www/ex1/tests_examples.tar
```

בדקו את תכניתכם וודאו שהפלטים שלכם זהים לאלה של פתרון בית הספר. אתם יכולים לייצר קבצי קלט רבים נוספים כדי לבדוק מקרים נוספים, ולהשוות את הפלט של התכנית שלכם עם פלטים של תלמידים אחרים, או עם הפלט שנוצר כשאתם נותנים את הקלט הזה לקובץ הריצה של פתרון בית הספר.

3. ביצוע overloading ב-C++:

http://www.cprogramming.com/tutorial/operator_overloading.html
http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B
http://www.cplusplus.com/reference/set/set/operators/

6. עבודה עם valgring:

1. בתרגיל זה (כמו ביתר התרגילים בקורס) תידרשו להשתמש בניהול זיכרון דינמי.
2. ישנו מבחר די גדול של תוכנות בשוק שמטרתם לסייע באיתור בעיות זיכרון בקוד לפני שחרורו אל הלקוח. אנו נשתמש בתוכנת valgrind, שיחסית לתוכנה חינמית, נותנת תוצאות מעולות.

3. כדי להריץ את valgrind עליכם לבצע קומפילציה ו-linkage לקוד שלכם עם הדגל '-g' (הן בשורת הקומפילציה והן בשורת ה-linkage). לאחר מכן הריצו valgrind:

```
> valgrind --leak-check=full --show-possibly-lost=yes
     --show-reachable=yes –undef-value-errors=yes  IntMatrixMainDriver
```

4. אם קיבלתם הודעת שגיאה, יתכן שתצטרכו לבצע שינוי הרשאות:

```
> chmod 777 IntMainDriver
```

5. כמובן שאם valgrind דיווח על בעיות עם הקוד שלכם, עליכם לתקן אותן.
6. היעזרו ב-tutorial הקצרצר של valgrind שבאתר הקורס.

7. הגשה:

1. עליכם להגיש קובץ tar בשם ex1.tar המכיל רק את הקבצים הבאים:
   - IntMatrix.h
   - IntMatrix.cpp
   - IntMatrixDriver.cpp
   - קובץ Makefile התומך בפקודות הבאות:
     - make IntMatrix- יצירת IntMatrix.o.
     - make IntMatrixMainDriver - קימפול ויצירת תוכנית IntMatrixMainDriver.
     - make tar - יצירת קובץ tar בשם ex1.tar, המכיל <u>רק</u> את הקבצים שצריך להגיש בתרגיל.
     - make all - ביצוע כל הפקודות - חוץ מ clean.
     - make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-makefile.
     - הרצת make ללא פרמטרים תבנה את קובץ ההרצה IntMatrixMainDriver
   - extension.pdf - <u>רק</u> במקרה שההגשה היא הגשה באיחור.
2. ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

```
tar cvf <tar_name> <files>
```

3. לפני ההגשה, פתחו את הקובץ ex1.tar בתיקיה נפרדת וודאו שהקבצים מתקמפלים ללא שגיאות וללא אזהרות.
4. מומלץ מאוד גם להריץ בדיקות אוטומטיות וטסטרים שכתבתם על הקוד אותו אתם עומדים להגיש.
5. בנוסף, אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:

```
~slabcpp/www/codingStyleCheck <file or directory>
```

כאשר <directory or file> מוחלף בשם הקובץ אותו אתם רוצים לבדוק או תיקייה שיבדקו כל הקבצים הנמצאים בה (שימו לב שבדיקה אוטומטית זו הינה רק חלק מבדיקות ה codingStyle)

6. דאגו לבדוק לאחר ההגשה את קובץ הפלט (submission.pdf) וודאו שההגשה שלכם עוברת את ה-presubmission script ללא שגיאות או אזהרות.

```
~slabcpp/www/ex1/presubmit_ex1
```

בהצלחה!