שפת C++ – תרגיל 2

תכנות מונחה עצמים. ירושה ופולימורפיזם. תכנון, STL.

תאריך הגשה: יום שני 4.1.16 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): יום שלישי 5.1.16 עד שעה 23:55

תאריך ההגשה של הבוחן: יום שני 4.1.16 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. עליכם לקמפל עם הדגלים Wall -Wextra ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה בציון התרגיל.** למשל, בכדי ליצור תוכנית מקובץ מקור בשם file.cpp יש להריץ את הפקודה:

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

6. עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי bit-64 (מחשבי האקווריום, 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

4. הנחיות ספציפיות לתרגיל זה:

1. בתרגיל זה הינכם רשאים (ואף נדרשים) להשתמש ב STL, לדוגמא map ו vector יכולים לעזור.
2. במידה והנכם משתמשים בהקצאות זיכרון דינמיות, הניחו כי הן מצליחות תמיד. מקובל להתמודד עם כישלון הקצאות זיכרון ב-C++ באמצעות מנגנון ה-exceptions אותו תתרגלו בתרגיל הבא.
3. בתרגיל זה אתם יכולים להניח כי קבצי הקלט תקינים, וכי הם יהיו בדיוק בפורמט שתואר בתרגיל.

5. מערכת אחזור מידע מוזיקלי - MIR (Musical Information Retrieval):

1. <u>רקע</u>

a) מערכות לאחזור מידע הן דבר נפוץ ומוכר – כגון מנועי חיפוש (גוגל למשל) או מערכת לחיפוש מאמרים בספריית האוניברסיטה. מערכות אלו מורכבות משרתים גדולים שמכילים מאגר מידע של פריטים רבים (למשל דפי web באינטרנט , או מסמכים טקסטואליים). המידע מסודר בצורה יעילה, המאפשרת חיפוש יעיל ומהיר.

b) הפעולה הנפוצה שהמערכת צריכה להתמודד איתה היא חיפוש (או שליפה) של פריט מתוך המאגר: השרת מקבל שאילתה (query) טקסטואלית – מילות חיפוש, והוא צריך להציג את תוצאות החיפוש בצורת רשימה מסודרת של הפריטים במאגר לפי סדר הרלוונטיות של הפריט למילות החיפוש (מהכי רלוונטי להכי פחות רלוונטי).

c) מערכות לאחזור מידע מוזיקלי הן מערכות לאחזור מידע שהפריטים במאגר שלהן הם שירים. עם קבלת מילות החיפוש המערכת מציגה את שמות השירים במאגר לפי הרלוונטיות שלהם למילות החיפוש. המאגר לא צריך לשמור את השירים במלואם ולרוב מסתפקים בשמירת מידע מצומצם שיהיה שימושי לצורך השליפה.

d) בתרגיל זה לא נעסוק בעיבוד אותות מוזיקליים, אלא נניח שכל המידע הדרוש על שיר חולץ כבר מראש (למשל באמצעות מאזינים או אלגוריתם לעיבוד מידע audio).

2. <u>יישום המערכת</u>

a) עליכם לכתוב תכנית בשם MIR (בין הקבצים צריך להיות קובץ הדרייבר, בעל פונקציה main, בשם MIR.cpp), המקבלת שלושה ארגומנטים, באופן הבא:

```
MIR <songs_data_file> <learned_parameters_file> <queries_file>
```

- הארגומנט הראשון הוא שם של קובץ המכיל את פרטי השירים שבמאגר, והמידע שידוע על כל אחד מהם (כאמור, אין צורך לשמור במאגר שירים שלמים, ובתרגיל זה, הקובץ הזה יכיל רק את הפרטים החשובים לצורך השליפה על כל שיר).
- הארגומנט השני הוא שם של קובץ המכיל את ערכי הפרמטרים של המערכת שנלמדו מראש, ומשפיעים על תהליך השליפה.
  (חלק חשוב בבניית מערכות MIR הוא מחקר מקדים ולמידה חישובית על סמך מדגמים פתורים[1]. בתרגיל אנחנו מניחים שתהליך הלמידה כבר בוצע ומסקנותיו מסוכמות בפרמטרים שבקובץ זה.)
- הארגומנט השלישי הוא שם של קובץ המכיל את מילות החיפוש (שאילתות).

b) בתרגיל אנו מניחים כי השירים מתחלקים לשני סוגים:

- שירים המכילים מילים

[1] מדגם פתור הוא רשימה של שירים, מילות חיפוש, ותוצאות חיפוש כלומר מידע הרלוונטיות של כל שיר לכל מילת חיפוש.

- שירים אינסטרומנטלים (ללא מלים)

c) פלט התכנית יהיה תוצאות השליפה ממאגר השירים לכל אחת ממילות החיפוש שבקובץ מילות החיפוש. לכל מילת חיפוש (לפי סדר הופעתה בקובץ מילות החיפוש) יוצג:

- שורה של קו הפרדה (ארבעים תווי מינוס)
- המילה
- שורה ריקה.
- רשימת כל שמות השירים מהמאגר, שקיבלו ציון חיובי (ציון > 0), מסודרים לפי ציון הרלוונטיות שלהם למילת החיפוש (מגבוה לנמוך).

לכל שיר יצוין הציון שלו ופריט מידע נוסף (מחבר המלים, אם זה שיר עם מלים, או המבצע, אם זה שיר אינסטרומנטלי).

לדוגמה:

```
----------------------------------------
Query word: cheerful

Turkish March\t14\tperformed by: the philharmonic orchestra
winter (the four seasons)\t13\tperformed by: London philharmonic
orchestra
take five\t12\tperformed by: The Dave Brubeck Quartet
The elephant\t6\tperformed by: The Dave Brubeck Quartet
Michelle\t2\tperformed by: Beatles unplugged
The turtle\t1\tperformed by: The Dave Brubeck Quartet
----------------------------------------
Query word: summertime

babe I'm gonna leave you\t10\tlyrics by: Anne Bredon
summertime\t5\tlyrics by: Gershwin - Heyward
```

(התו '\t' הוא <tab>).

הסתכלו בקבצי הפלט לדוגמה, והשתמשו בפתרון בית הספר בכדי להתאים את הפורמט במדויק.

בכל מקרה של סתירה בין הנכתב כאן לפלט פתרון בית הספר, יש להתאים את הפלט שלכם לפתרון בית הספר.

3. <u>אלגוריתם השליפה</u>

a) על כל מילת החיפוש המערכת תעבור על כל השירים במאגר ותחשב לכל אחד מהם ציון (מספר שלם int המציין את רמת הרלוונטיות של השיר למילת המפתח), לאחר מכן המערכת תמיין את השירים (שהציון שלהם חיובי) לפי הציון ותציג אותם מגדול לקטן.

b) ציון השיר מורכב מסכימה של 4 תתי הציונים הבאים:

1) <u>ציון התאמת-תגיות</u> - במידה ומילת החיפוש מופיעה כאחת מהתגיות[2] שניתנו לשיר:
(עוצמת-התגית[3]) * (משקל-התאמת-תגיות[4])

2) <u>ציון התאמת-מילים</u> - במידה והשיר מכיל מילים:
(מספר הפעמים בהם מופיעה מילת החיפוש ב- מילות-השיר) * (משקל-התאמת-מילים[5])

3) <u>ציון התאמת-כלי-נגינה</u> - במידה והשיר הוא אינסטרומנטלי, ובמידה והמילה מופיעה ברשימת כלי הנגינה של השיר:
משקל-התאמת-כלי-נגינה[6].

4) <u>ציון התאמת-קצב</u> - במידה ומתקיימים התנאים הבאים:
- השיר הוא אינסטרומנטלי
- בפרטי השיר צוין גם קצב השיר - bpm
- מילת החיפוש היא מילה ידועה מראש (כלומר למילה זו התפלגות נורמלית של ערכי bmp) עם ממוצע m וסטיית תקן s

משקל-התאמת-קצב מוכפל בעיגול כלפי מטה של האקספונט הבא[7]:

$$e^{-\frac{(bpm-m)^2}{2s^2}}$$

(מוטיבציה:
ישנן מילות חיפוש שמתארות מושג מופשט (למשל מילות חיפוש של רגשות), להן יש חשיבות לקצב של השיר. לדוגמה אם מחפשים שיר עצוב (מילת חיפוש "sad") סביר יותר ששירים רלוונטיים יהיו שירים איטיים יותר (כלומר עם bpm נמוך).
המודל של המערכת שלנו מניח כי למילת חיפוש כזו יש התפלגות נורמאלית של ערכי bpm (עם ממוצע m וסטיית תקן s שניתנים בקובץ הפרמטרים), כך שאפשר לבחון את הנראות[8] (likelihood) של שיר עם ערך bpm מסוים בהינתן מילת חיפוש כזו.
(הקשר בין בין מילות חיפוש קונספטואליות נפוצות לבין קצבים של שירים, גם הוא נלמד מראש.))

לדוגמא:
עבור הקבצים parameters_1.in, songs_1.in (שמופיעים בתקיית התרגיל)
יבוצעו החישובים הבאים:

```
song:babe I'm gonna leave you
query:blues
      TagScore: 50
```

---

[2] התגיות עשויות להיות מוגדרות הן באמצעות מערכת תיוג אוטומטית והן באמצעות הזנה ידנית. והן עשויות להיות מכל מיני סוגים - כלי נגינה שמופיעים בשיר, סגנון, רגש המובע בשיר וכד'.
[3] כל תגית מורכבת משם התגית (מחרוזת) ועוצמתה (ערך מספרי מסוג int).
[4] מתקבל באמצעות קובץ הפרמטרים - ראו להלן.
[5] מתקבל באמצעות קובץ הפרמטרים - ראו להלן.
[6] מתקבל באמצעות קובץ הפרמטרים - ראו להלן.
[7] ציון-התאמת-קצב * floor[e^(-((bpm-m)^2)/(2*(s^2)))]
[8] ההסתברות לקבל ערך bpm כזה, בהינתן שזהו שיר שרלוונטי למילת החיפוש.

```
      LyricSong: 0
      instruments: -
      BpmLikelihoodScore: -
      score:50
++++++++++++++++++++++
song: babe I'm gonna leave you
query:babe
      TagScore: 0
      LyricSong: 80
      instruments: -
      BpmLikelihoodScore: -
   score:80
++++++++++++++++++++++
song:take five
query:cheerful
      TagScore: 0
      LyricSong: -
      instruments: 0
      getBpmLikelihoodScore: 38
      score:38
```

c) המיון:

- את מיון השירים לפי רלוונטיות אתם יכולים לבצע באיזו דרך שתרצו. אתם רשאים לממש מיון שסיבוכיות זמן הריצה שלו ריבועית במספר השירים במאגר n: O(n^2).
- **שימו לב**:
  - עליכם להשתמש באלגוריתם "מיון יציב". כלומר, במקרה שיש כמה שירים שקיבלו אותו ציון (לא אפס), סדר הצגתם צריך להיות בהתאם לסדר המקורי שלהם במאגר השירים (הסדר שבו הופיעו בקובץ השירים).
  - הקפידו לממש את המיון בפונקציה נפרדת (אם תרצו, אפילו בקובץ נפרד), כדי ששאר הקוד לא יהיה מושפע באופן ביצוע המיון.
  - לרשותכם קבצי עזר SortHelper.h ו SortHelper.cpp. באמצעותם אפשר להכניס ערכי ציון (מספרים שלמים) לרשימה אחד אחרי השני (לפי הסדר המקורי של השירים במאגר) וכלי העזר יימיין אותם מהגדול לקטן (תוך שמירה על הסדר המקורי של שירים עם אותו ציון) וייתן את הסדר בו צריך לעבור על האינדקסים המקוריים של השירים במאגר כדי לסדר אותם מהגדול לקטן. אתם מוזמנים להשתמש בקבצי עזר אלה, כרצונכם.

4. <u>פורמט קבצי הקלט</u>

a) קובץ פרטי השירים:

- כל שיר מתחיל בשורה עם התו '='.
- בשורה הבאה מופיע שם השיר.
- בשורה הבאה התגיות שיש לשיר: כל תגית מורכבת ממילה אחת רווח ומספר שלם(מטיפוס int) התגיות מופרדות ביניהן ברווחים. והרשימה מוקפת בסוגריים מסולסלים.

- בשורה הבאה או מילות השיר או רשימת כלי נגינה בשיר (אם הוא אינסטרומנטלי) - **אבל לא שניהם**.
- בשורה הבאה או שם מחבר המלים (אם זה שיר עם מלים) או שם המבצע/ים (אם זה שיר אינסטרומנטלי - **אבל לא שניהם**.
- אם זהו שיר אינסטרומנטלי, עשויה להיות שורה נוספת המכילה את ה bpm (מספר ממשי מטיפוס double.
- לאחר השיר האחרון תופיע שורה המכילה שלוש כוכביות "***".

לדוגמה:

```
=
title: yonatan hakatan
tags: {kids 1 folk 4 cheerful 3}
lyrics: {yonatan hakatan ratz baboker el hagan}
lyricsBy: Amami
=
title: take five
tags: {jazz 1 relax 1 dancing 1 light 1}
instruments: {saxophone trumpet drums guitar bass}
performedBy: Jazzi McJazz bigband
bpm: 90.3
***
```

b) קובץ הפרמטרים:

- קובץ הפרמטרים יפרט את משקל-התאמת-תגיות, משקל-התאמת-מלים, משקל-התאמת-כלי-נגינה, משקל-נראות-קצב (ארבעתם מספרים שלמים וכל אחד מהם בשורה נפרדת)
- לאחר מכן תופיע רשימה של מילות חיפוש ידועות מראש והממוצע וסטיית התקן של bpm לכל מילה כזו. לכל מילה כזו תהיה שורה אחת עם המילה ושני ערכים מספריים ממשיים (מופרדים באמצעות פסיק ורווחים), ממוצע bpm וסטיית תקן bpm, בהתאמה.

לדוגמה:

```
tagMatchScore: 50
lyricsMatchScore: 35
instrumentMatchScore: 35
bpmLikelihoodWeight: 55
sad: 30.123 , 9.87
cheerful: 97.65 , 30.1
anger: 110.3 , 25.45
```

c) קובץ מילות החיפוש (השאילתות):

- בכל שורה תופיע מילת חיפוש (מחרוזת של מילה אחת).

לדוגמה:

```
sad
anger
```

d) שימו לב! :

- לכל אורך התרגיל אתם יכולים להניח כי קבצי הקלט תקינים (ואינכם צריכים להתמודד עם קבצים בפורמט לא תקין. אם כי קבצים המכילים שורות ריקות בסוף הקובץ הם קבצים בפורמט תקין)
- לכל אורך התרגיל ערכי מספרים תמיד יהיו חיוביים ממש (>0).

5. <u>תכנון התכנית:</u>

a) זהו תרגיל בתכנון (design) ובמימוש. יש לכם חופש לתכנן את חלוקת המחלקות, תפקידיהן והקשרים ביניהן.

b) כשיש כמה אפשרויות, תצטרכו לשקול יתרונות וחסרונות של כל אחת ולבצע בחירה לפי שיקול דעתכם. אבל זה גם אומר שלבודקים יהיה מקום לשיקול דעת. חופש בתכנון לא אומר שכל תכנון הוא טוב. עדיין יכולות להיות החלטות תכנוניות גרועות או "שגויות", לא חכמות. לכן עליכם לחשוב ולתכנן את מבנה המחלקות היטב (ממולץ על הנייר) לפני שתתחילו לממש את הקוד.

c) שמות המחלקות, המשתנים והפונקציות צריכים להיות אינפורמטיביים ובעלי משמעות לפי תפקידם.

d) תעדו בבירור את הקוד שלכם.

e) חישבו היטב כיצד להשתמש ב const, אילו רכיבים (מתודות, משתנים) צריכים להיות סטטיים, אילו מחלקות מייצרות מופעים ואילו לא (ולכן צריכות להיות אבסטרקטיות)

f) חישבו אילו מתודות צריכות להיות מוגדרות כ pure virtual.

g) חישבו אילו בנאים צריכים להיות ציבוריים ואילו צריכים להיות חבויים.

h) הוסיפו בקובץ README תיאור קצר של תכנון התכנית שלכם (אילו מחלקות ישנן, מה מבנה הירושה. מה הקשרים בין המחלקות), והסבר קצר על החלטות שביצעתם, למה בחרתם לתכנן כך.

6. <u>טיפים והנחיות:</u>

a) לרשותכם קובץ עזר ReadingSongsTemplate.cpp. אתם מוזמנים לקרוא ולהתרשם ממנו או אפילו להשתמש בו בתכניתכם. הקובץ לא שלם (ובטח לא יתקמפל בגרסתו הנוכחית), והוא דורש שינויים ותוספות שיתאימו למחלקות שאתם תגדירו. השימוש בו <u>אינו חובה</u> ומטרתו להקל עליכם בקריאת השירים מקובץ השירים (במידה ואתם משתמשים בו עליכם כמובן לצרף אותו בהגשה. רצוי גם לשנות לו את השם למשהו מתאים יותר). <u>שימו לב</u> שהקוד בקובץ זה לא כתוב בהתאם לנהלי הקורס ובמידה ואתם משתמשים בו עליכם לשכתב אותו בהתאם (כולל פירוק מתודות למתודות משנה, במידה ויש צורך בכך).

b) קריאת הקבצים היא לא החלק החשוב של התרגיל ואנחנו לא רוצים שתשקיעו יותר מדי זמן בזה. החלק החשוב הוא תכנון (design) התכנית (חלוקה למחלקות, תכנון ירושה) ושימוש בירושה ובפולימורפיזם. כדאי שקודם תשקיעו בתכנות ובבניית המחלקות, ורק אז תפנו לחלק של קריאת הנתונים מהקבצים (רק אז תדעו מה לעשות עם הנתונים שתקראו – אילו אובייקטים לייצר וכדו').

c) כדי למנוע חופש מוחלט, וכדי לתת לכם מעט מסגרת **<u>נחייב</u>** כמה הגבלות:

- זהו תרגיל שמטרתו לימוד ירושה ופולימורפיזם. חובה עליכם להשתמש בירושה.

- על התכנית לקרוא את המידע (פרטי השירים מהמאגר והפרמטרים של המערכת) ולאתחל מערכת עם כל המאגר בזיכרון, ורק לאחר מכן יש לטפל בשאילתות שבקובץ מילות החיפוש.
- יש לטפל בכל שאילתה בנפרד.
- גודל המאגר (כמות השירים) וכמות השאילתות (מילות החיפוש) איננה ידועה מראש.
- מגבלה על הזיכרון: בכל רגע נתון הזיכרון השמור בתכנית יהיה ליניארי במספר השירים במאגר (ובמספר מילות החיפוש הקונספטואליות הידועות מראש מקובץ הפרמטרים), ולא יהיה תלוי במספר השאילתות בקובץ מילות החיפוש.

6. חומר עזר:

1. את פתרון הבית ספר ניתן למצוא ב:

```
~labcpp/www/ex2/schoolSol.tar
```

2. את קבצי העזר ניתן למצוא ב:

```
~labcpp/www/ex2/ex2_files.tar
```

שימו לב שאין חובה להשתמש בקבצים אלו והם נועדו להקל עליכם.

3. דוגמאות לטסטים ניתן למצוא ב:

```
~labcpp/www/ex2/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.

7. עבודה עם valgrind:

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.

8. הגשה:

1. עליכם להגיש קובץ tar בשם ex2.tar המכיל <u>את כל הקבצים הנדרשים לקימפול התוכנית שלכם</u> ואת הקבצים הבאים:
   - README
   - קובץ Makefile התומך בפקודות הבאות:
     - make - קימפול ויצירת תוכנית MIR.
     - make tar - יצירת קובץ tar בשם ex2.tar, המכיל <u>רק</u> את הקבצים שצריך להגיש בתרגיל.
     - make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-makefile.
   - extension.pdf - <u>רק</u> במקרה שההגשה היא הגשה באיחור.

   שימו לב! - אל אף שאתם יכולים להוסיף קבצים נוספים כרצונכם, המנעו מהוספת קבצים לא רלוונטים (גם בכדי להקל על הבודקים, וגם בכדי שציונכם לא יפגע מכך).
2. ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

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

3. לפני ההגשה, פתחו את קובץ ה-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:

```
~labcpp/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.

```
~labcpp/www/ex2/presubmit_ex2
```

בהצלחה!