# תרגיל 1 – slabcpp

## היכרות עם השפה: אובייקטים, בנאים, הורסים, heap/stack, העברה ע"י reference, operator overloading

תאריך הגשה: יום חמישי, 28.8.2014 עד שעה 23:55
תאריך הגשה מאוחרת (בהפחתה של 10 נקודות): אין

### הנחיות חשובות (בעיקר לפני הגשת התרגיל הראשון)

- עליכם לקרוא את הנחיות הגשת התרגילים ומדיניות הקורס. כמו כן עליכם לקרוא את ההנחיות לסגנון כתיבת קוד. שני המסמכים נמצאים באתר הקורס והינם חשובים במיוחד מאחר והניקוד בתרגילים יכלול גם עמידה בהנחיות אלו.
- אין להגיש קובץ README אלא אם צוין במפורש שיש צורך בכך (לדוגמא, בתרגיל זה אין צורך להגיש קובץ README)
  - אין לענות בקובץ ה-README על שאלות שנשאלו בתרגיל באופן מפורש, אלא במקומות שהוגדרו לכך (לדוגמא, בתיעוד הקוד).
- עליכם לקמפל עם הדגל "Wall–" על מנת לוודא שתכניתכם מתקמפלת ללא אזהרות.
  - תכנית שמתקמפלת עם אזהרות תגרור הורדת נקודות בציון התרגיל
  - לדוגמא, כדי לקמפל קובץ מקור בשם **ex1.cpp** לקובץ ריצה בשם **ex1** יש להשתמש בפקודה:

```
> g++ -Wall ex1.cpp -o ex1
```

- עליכם לוודא שהתרגילים שלכם תקינים ועומדים **בכל דרישות הקימפול והריצה במחשבי בית הספר** המבוססים 64-bit (מחשבי האקווריום, לוי, השרת river וכו'). מומלץ לקמפל ולהריץ את התרגיל במחשבי בית הספר לפני ההגשה.
- לאחר ההגשה, בדקו את הפלט המתקבל בקובץ ה-PDF שנוצר מה-presubmission script בזמן ההגשה. באם ישנן שגיאות, תקנו אותן והגישו שוב על מנת שלא לאבד נקודות.
- בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם.
  - חישבו על מקרי קצה לבדיקת הקוד, חלק מהציון ניתן על עמידה בבדיקות אוטומטיות.
  - קבצי בדיקה לדוגמא שמסופקים על ידי צוות הקורס אינם בהכרח נבדקים על ידי הבדיקה האוטומטית בהגשה (presubmission script). שימוש בקבצי הבדיקה לדוגמא הוא באחריותכם.
- ניתן להתחבר באמצעות SSH למחשבי בית הספר (למשל לשם בדיקת הקוד במחשבי בית הספר לפני הגשה מהבית)

http://wiki.cs.huji.ac.il/wiki/Connecting_from_outside

### הנחיות חשובות לכלל התרגילים בקורס C++

- הקפידו להשתמש בפונקציות של C++ (למשל new, delete, cout) על פני פונקציות של C (למשל malloc, free, printf). בפרט השתמשו במחלקה string (ב-header: string) ולא במחרוזת של C (char *).
- יש להשתמש בספריות סטדנרטיות של C++ ולא של C אלא אם כן הדבר הכרחי.
- הקפידו על עקרונות Information Hiding – לדוגמא, הקפידו כי משתני המחלקות שלכם מוגדרים כמשתנים פרטיים (private).
- הקפידו לא להעתיק by value משתנים כבדים, אלא להעבירם (היכן שניתן) by reference.

- **הקפידו מאוד** 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.
- הקפידו לשחרר את כל הזיכרון שאתם מקצים (השתמשו ב-valgrind כדי לבדוק שאין לכם דליפות זיכרון).

## שאלה 1 (35 נקודות)

בשאלה זאת נבנה רשימה מקושרת.
את הרשימה עליכם לממש בקבצים MyLinkedList.h ו-MyLinkedList.cpp.
הרשימה תהיה רשימה מקושרת דו כיוונית של מחרוזות ומספרים ממשיים (כל צומת יחזיק מחרוזת (key) ומספר ממשי (data)).
בנוסף לבנאי והורס עליכם לממש את השיטות הבאות:

- המתודה add תוסיף איבר (מחרוזת ומספר) לסוף הרשימה.
- המתודה remove תקבל מחרוזת. היא תוציא את כל האיברים ברשימה המכילים מחרוזת זאת. המתודה תחזיר את מספר האברים שהוסרו.
- המתודה isInList מקבלת שני פרמטרים: מפתח (מחרוזת) ו reference ל- double. המתודה מחזירה ערך בוליאני true אם המפתח מופיע ברשימה (לפחות פעם אחת) ו- false אחרת. אם המפתח הופיע ברשימה היא תעדכן את המשתנה של ה- reference להיות ה- data של המפתח שנמצא (אם קיימים מספר מפתחות מתאימים, יוחזר ה- data של אחד מהם, לא משנה איזה).
ראו קריאה למתודה בקובץ ListExample.cpp.
אין הגבלת סיבוכיות על מתודה זו, ניתן לממשה ב O(n) כאשר n – מספר האיברים ברשימה.
- המתודה printList תדפיס את כל איברי הרשימה. כל שורה תחיל זוג ערכים המופרדים ע"י פסיק יחיד, ללא רווחים:

```
<string value>,<double value>
```

בלבד. אם הרשימה ריקה, תודפס המילה Empty. זכרו בסוף כל הדפסת רשימה (בין אם היא מכילה מחרוזת ובין אם היא מכילה את המילה Empty) להדפיס את תו הזנת שורה ('\n').
- המתודה sumList מחזירה את סכום המספרים הממשיים ברשימה (סכום כל ה- data של אברי הרשימה).
- נתון לכם קובץ בדיקה ListExample.cpp וקובץ הפלט הצפוי ListExample.sol. הקובץ מבצע בדיקות בסיסיות. באחריותכם לבצע בדיקות נוספות ומעמיקות יותר.

**הנחיות נוספות**

- יש לממש את מתודת הוספת איבר בצורה יעילה. סיבוכיות זמן הריצה של המתודה add צריכה להיות O(1).
- וודאו שהקוד שלכם עובד עם קובץ הדוגמא
~slabcpp/public/ex1/code_files/ListExample.cpp
הפלט שהקובץ אמור לייצר מופיע ב-
~slabcpp/public/ex1/inputOutput/ListExample.sol
- כלל קבצי התרגיל ניתנים גם להורדה מאתר הקורס
ex1_files.tar
- ניתן לממש את הקוד ע"י מחלקה יחידה. ניתן (אך לא חובה) להשתמש במחלקה נוספת (MyLinkedListNode) שתייצג איבר ברשימה. אם בחרתם לממש את המחלקה MyLinkedListNode, הקפידו להגיש אותה. שימו לב כי ניתן לממש את MyLinkedListNode כולה (כולל הצהרת ה-class) בקובץ MyLinkedList.cpp, ולא לאזכר את קיום המחלקה בקובץ ה-header (כך יותר נכון מבחינת information hiding).

- טיפול בשגיאות ב-C++ מבוצע ע"י מנגנון חריגות (exceptions). אתם תלמדו על הנושא בהמשך הקורס. בשיעורי בית אלו אתם רשאים להניח כי אין שגיאות (הקצאות זיכרון מצליחות תמיד, הפרמטרים לפונקציות חוקיים וכו').
בפרט, הפעולה הוספת איבר לרשימה מצליחה תמיד (אחרי כל קריאה לפונקציה של הוספת איבר לרשימה, גודל הרשימה יגדל ב-1).
- נתון לכם קובץ בדיקה ListExample.cpp וקובץ הפלט הצפוי ListExample.sol. הקובץ מבצע בדיקות בסיסיות. באחריותכם לבצע בדיקות נוספות ומעמיקות יותר.

## שאלה 2 (30 נקודות)

בשאלה זאת נממש HashMap – מבנה נתונים הממפה ביעילות בין מחרוזות (מפתחות) למספרים ממשיים (data) באמצעות hash table. אנו נממש את מבנה הנתונים בעזרת הרשימה המקושרת מהשאלה הקודמת.
HashTable מבוסס רשימות מקושרות הוא מבנה נתונים יעיל מאוד למימוש מיפויים בין ערכים, הנפוץ מאוד בתעשייה.

**המשימה שלכם**
הפעם אנו נותנים לכם את קובץ ה-interface שעליכם לממש:
MyHashMap.h
ממשו אותו בקובץ MyHashMap.cpp.

כל אובייקט של המחלקה MyHashMap מכיל מערך של רשימות מקושרות (בגודל הקבוע HASH_SIZE). כאשר מוסיפים למבנה הנתונים מיפוי חדש בין מחרוזת למספר ממשי, אינדקס הרשימה המקושרת שבה יוחזק המיפוי החדש נקבע באמצעות פונקציית hash. בתרגיל זה זוהי פונקציה סטטית של המחלקה MyHashMap ושמה יהיה myHashFunction.
המחלקה MyHashMap תכיל גם מספר פונקציות סטנדרטיות עבור Hash table אותן ניתן לראות בממשק הנתון.

**הנחיות**
- ייתכן שלצורך מימוש הפונקציה isIntersect תרצו להוסיף מתודות ציבוריות (public) נוספות לקובץ MyLinkedList.h. אתם חופשיים להוסיף מתודות (הן ל- MyLinkedList.h והן ל- MyHashMap.h) ציבוריות או פרטיות. באופן דומה, אתם חופשיים להוסיף משתנים למחלקות הללו. כמובן שעליכם להקפיד לממש את כל המתודות שהתבקשתם בתיאור התרגיל.
- אין להשתמש במבני נתונים מוכנים (כמו STL) לצורך פתרון התרגיל.
- מחרוזת ריקה היא מחרוזת חוקית לפונקצית ה-hash שלנו (ערך פונקצית ה hash עבורה היא 0).
- בשאלה זו לא נבדוק את סיבוכיות המתודות, אך עליכם לציין זאת כחלק מהתיעוד.
- שימו לב לדרישת פונקצית הוספת איבר של ה- interface – אם המפתח כבר קיים ב-HashMap, אנו רק מחליפים עבורו את ה- data המתאים. בפרט, ה-interface לא מאפשר ל-HashMap להחזיק בשלב כלשהו שני איברים עם keys (מחרוזות) זהים. דבר זה שונה מהגדרת הרשימה המקושרת בשאלה הקודמת (שיכלה להחזיק מספר node-ים שונים עם אותה המחרוזת).
- נתון לכם קובץ בדיקה HashSimpleCheck.cpp וקובץ הפלט הצפוי HashSimpleCheck.sol. הקובץ מבצע בדיקות בסיסיות. באחריותכם לבצע בדיקות נוספות ומעמיקות יותר.

## שאלה 3 (35 נקודות)

בשאלה זו עליכם לממש את האופרטורים הבאים עבור המחלקה MyHashMap :

1. אופרטורים להשוואה:
   a. operator<
   b. operator>
   c. operator==

   אופרטורים אלו יערכו את ההשוואות בין שתי HashMaps כאשר הערך הקובע בהשוואה הוא משקלן (כפי שמוחזר מהפונקציה totWeight שהוגדרה בחלק הקודם).
   ערך ההחזרה של האופרטורים הוא bool

2. operator -
   מאחר ו-MyHashMap הינו Set (אף מפתח אינו מופיע יותר מפעם אחת), אופרטור זה יתנהג כמו חיסור על Set. הפונקציה תחזיר HashMap חדשה המכילה את האיברים מה-HashMap השמאלי אשר המפתחות שלהם אינם מופיעים ב-HashMap הימני. זוהי הפעולה המתמטית של חיסור על sets הנכתבת באופן הבא: left\right.

3. operator |
   זהו אופרטור המבצע את פעולת ה-Union על sets. הפונקציה תחזיר HashMap חדשה המכילה את כל האיברים מה-HashMap השמאלי וכן את כל האיברים מה-HashMap הימני אשר אינם מופיעים ב-HashMap השמאלי. זאת אומרת שאם היו במקור איברים עם מפתח שמופיע גם ב-left וגם ב-right, אז ב-HashMap החדשה הערך של המפתחות הללו יהיה הערך שהופיע ב-left והערכים של מפתחות מ-right שלא היו ב-left כלל ישמרו. זוהי הפעולה המתמטית של איחוד על sets הנכתבת באופן הבא: left ∪ right

4. operator &
   זהו אופרטור המבצע את פעולת ה-Intersection על sets. הפונקציה תחזיר HashMap חדשה המכילה את כל האיברים שמפתחותיהם מופיעים גם ב-HashMap השמאלי וגם בימני. הערכים שישמרו במפה החדשה יהיו הערכים שהיו ב-HashMap השמאלי. זאת אומרת שאם היו במקור איברים עם מפתח שמופיע גם ב-left וגם ב-right, אז ב-HashMap החדשה הערך של המפתחות הללו יהיה הערך שהופיע ב-left בלבד. זוהי הפעולה המתמטית של חיתוך על sets הנכתבת באופן הבא: left ∩ right

5. Copy Constructor
   את בנאי ההעתקה, יהיה עליכם להעביר מ-private ל-public. חתימת הפונקציה הינה:
   `MyHashMap(const MyHashMap&);`
   פונקציה זו צריכה להיות ממומשת בכדי למלא את דרישות חלק 3 בתרגיל ולתמוך בהשמה של תוצאות המתקבלות מהאופרטורים האחרים אותם עליכם לממש.
   שימו לב לממש את הפונקציה כך שהעותק החדש יהיה עצמאי מזה שהתקבל כפרמטר. לדוגמא, בעת מחיקת המפה המקורית וזו שהועתקה, שימו לב שלא יהיו מקרים של מחיקת פוינטר זהה פעמיים.

עליכם יהיה לכתוב גם קובץ בשם HashMapBinaryOperations.cpp המכיל main ויוצר קובץ הרצה המדגים את השימוש בכל ששת האופרטורים שהוגדרו כאן.

**הנחיות**

- יש לממש את האופרטורים במחלקה עצמה בקובץ MyHashMap.h ו-MyHashMap.cpp כפונקציות מחלקה. הבנת החתימות הנדרשות של פונקציות האופרטורים הן חלק מהדרישות.

- בשאלה זו אין צורך לממש מחלקה כלשהי, אלא רק לממש את המתודות המבוקשות (ניתן לכתוב גם מתודות עזר).
- מותר לכם להוסיף מתודות למחלקות MyHashMap ו-MyLinkedList מלבד האופרטורים.
- הקפידו שהפונקציות תעבודנה ללא דליפות זיכרון.
- הקפידו שלא לשכפל קוד בין האופרטורים השונים.
- נתון לכם קובץ בדיקה TestHashMapBinFuncs.cpp וקובץ הפלט הצפוי TestHashMapBinFuncs.sol. הקובץ מבצע בדיקות בסיסיות. באחריותכם לבצע בדיקות נוספות ומעמיקות יותר.

## הוראות הגשה

עליכם להגיש:

- קובץ Makefile התומך במטרות הבאות:
  - הרצת make ללא פרמטרים תבנה את קובץ ההרצה HashMapBinaryOperations מהקבצים
    - HashMapBinaryOperations.cpp
    - MyHashMap.h
    - MyHashMap.cpp
    - MyLinkedList.h
    - MyLinkedList.cpp
  - הרצת make clean תמחק קבצים זמניים, קבצי אובייקט וקבצי ריצה.
  - ניתן ורצוי לתמוך גם במטרות נוספות כגון all הבונה את כלל המטרות ו-test המריצה מספר בדיקות אוטומטיות

הקומפילציה צריכה להתבצע ללא אזהרות (warnings) או שגיאות.
עליכם להשתמש בקומפיילר g++ המותקן במחשבי בית הספר. זכרו להשתמש בדגל Wall.
דוגמא לביצוע קומפילציה:

```
> g++ -Wall MyLinkedList.cpp MyHashMap.cpp
HashMapBinaryOperations.cpp -o HashMapBinaryOperations
```

- יש להגיש את הקבצים המממשים את הפתרונות לשאלות 1-3 :
  - HashMapBinaryOperations.cpp
  - MyHashMap.h
  - MyHashMap.cpp
  - MyLinkedList.h
  - MyLinkedList.cpp

כאמור, מותר לכם להרחיב את הקובץ MyHashMap.h, אך וודאו כי מימשתם את כל ה-interface המקורי שסיפקנו לכם וכן את האופרטורים המבוקשים בשאלה 3.

צרו קובץ tar להגשה ע"י הרצת הפקודה הבאה מה-shell:

```
> tar cvf ex1.tar HashMapBinaryOperations.cpp MyHashMap.h
MyHashMap.cpp MyLinkedList.* Makefile
```

## בדיקה

לרשותכם כמה קבצי בדיקה וקבצי פלט של פתרון בית הספר:

```
~slabcpp/public/ex1/inputOutput/
```

קבצים אלו כלולים גם בקבצי התרגיל:

ex1 files.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.

בדקו את תכניתכם מבחינת סגנון הקוד. וודאו שהקוד עומד בסטנדרטים של הקורס.

דאגו לבדוק שקובץ ההגשה שלכם עובר את ה presubmission script ללא שגיאות וללא אזהרות:

~slabcpp/public/ex1/presubmit_ex1 ex1.tar

**סיכום הגשה**

עליכם להגיש קובץ tar בשם ex1.tar שמכיל את הקבצים הבאים:

- HashMapBinaryOperations.cpp
- MyHashMap.h
- MyHashMap.cpp
- MyLinkedList.h
- MyLinkedList.cpp
- Makefile

ניתן להגיש גם את הקבצים:

- MyLinkedListNode.h
- MyLinkedListNode.cpp

**קישורים**

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/

בהצלחה!