

תרגיל 4

הנחיות חשובות לכלל התרגילים מעתה והלאה בקורס:

- התרגילים הם לעבודה ביחידים. מותר להתייעץ אך ורק בעל פה, אסור בתכלית האיסור שחומר כתוב/מודפס/אלקטרוני יעבור בין אנשים. בנוסף, על חלק מהתרגילים תיבחנו פרונטלית ועליכם להבין כל דבר בקוד!
- המנעו ממספרי קסם: מספרים שמופיעים באמצע הקוד בלי משמעות מיוחדת (לדוגמה נניח שמספר הרשומות בתרגיל אחר הוא מקסימום 50 ואז בכל מקום בקוד כתוב 50. לעומת זאת, 0 לתחילת מערך לא נחשב מספר קסם - הפעילו הגיון בריא) והשתמשו במקום זאת בפקודות מאקרו (`#define` או אם כבר למדתם על כך ב `const`).
- אין להשתמש ב-`variable length arrays`, וכן במשתנים סטטיים `static` / או גלובליים.
- כל התרגילים בקורס צריכים להתקמפל ולרוץ באתר `c9.io` (האתר מריץ מערכת הפעלה אובונטו) עם שורת הקמפול:

```
gcc -Wall -Wvla -Werror -g ...
```

ושורת הקמפול:

```
g++ -Wall -Wvla -Werror -g -D_GLIBCXX_DEBUG -std=c++11 ...
```

עבור תרגילי C++

או במידה ומצורף Makefile עם ה `Makefile` המצורף

- יש להקפיד על סגנון תכנות טוב כמו שלמדתם. לדוגמה, להימנע מחזרות קוד (לכתוב פונקציות שצריך), שמות משתנים עם משמעות, בהירות הקוד, להקפיד להשתמש בקבועים שצריך ולא במספרי קסם וכו'.
- עליכם להגיש קובץ ששמו מספר ת.ז. שלכם כמו שהיא מופיעה באתר המודל נקודה `zip` . לדוגמה, אם מספר ת.ז. שלי הוא 12345678 אז שם הקובץ יהיה:
`12345678.zip`
- הקפידו להשתמש בפונקציות של C++ (למשל `new`, `delete`, `cout`) על פני פונקציות של C (למשל `malloc`, `free`, `printf`). בפרט השתמשו במחלקה `string` ולא במחרוזת של C (כלומר, `char*`).
- יש להשתמש בספריות סטנדרטיות של C++ ולא של C אלא אם כן זה הכרחי.
- הקפידו על עקרונות `Information hiding`. לדוגמה, הקפידו להגדיר משתני מחלקות כ `private`.
- הקפידו לא להעביר אובייקטים גדולים `by value` אלא `by reference` או `by const reference`.
- הקפידו מאד על `const correctness`. כלומר, על שימוש במילה השמורה `const` בצורה נכונה.
- אם אתם מקצים זיכרון דינמי, הקפידו לשחרר אותו (מומלץ להשתמש ב `valgrind` על מנת לוודא שאין דליפות זיכרון).

תרגיל:

בתרגיל זה נבנה רשימה מקושרת.

את הרשימה עליכם לממש בקבצים MyLinkedList.hpp (הצהרות בלבד) ו- MyLinkedList.cpp (מימוש). הרשימה תהיה רשימה מקושרת **דו כיוונית** של מחרוזות ומספרים ממשיים (כל צומת יחזיק מחרוזת (key, ייצגו ע"י std::string) ומספר ממשי (data)).
יהי n מספר איברים ברשימה, n₁, n₂ מספרי איברים ברשימות כאשר יש 2 רשימות. אפשר להניח שאורך כל מחרוזת key הוא O(1).

עליכם לממש:

- בנאי חסר פרמטרים שיוצר רשימה ריקה. סיבוכיות נדרשת: O(1)
- בנאי המקבל 2 מערכים פרימיטיביים ואורכם (2 המערכים באורך שווה) ומייצר רשימה המכילה בזוגות את ה keys וה values (ראו גם שימוש בטסט שניתן). סיבוכיות נדרשת: O(n)
- בנאי מעתיק היוצר עותק עמוק (deep copy). סיבוכיות נדרשת: O(n)
- הורס (destructor) המנקה את כל הזיכרון בו השתמשתם. סיבוכיות נדרשת: O(n)
- אופרטור השמה (=operator) המנקה את צד שמאל ויוצר עותק עמוק של צד ימין בצד שמאל. סיבוכיות נדרשת: O(n₁+n₂)
- מתודה add המוסיפה איבר **לסוף הרשימה**. סיבוכיות נדרשת: O(1)
- מתודה remove שמקבלת מחרוזת ומוציאה את **כל** האיברים ברשימה המכילים את מחרוזת זאת. המתודה תחזיר את מספר האיברים שהוצאו. סיבוכיות נדרשת: O(n)
- המתודה isInList מקבלת 2 פרמטרים: מפתח (מחרוזת) ומספר ממשי. המתודה מחזירה ערך בוליאני true אם ה**צירוף** של המפתח והמספר מופיע ברשימה לפחות פעם אחת ו- false אחרת. סיבוכיות נדרשת: O(n)
- המתודה sumList מחזירה את סכום המספרים הממשיים ברשימה (סכום כל ה- data של איברי הרשימה). סיבוכיות נדרשת: O(n)
- מתודה size המחזירה את גודל הרשימה. סיבוכיות נדרשת: O(1)
- אופרטור == ואופרטור != המשווים לוגית בין הרשימה לרשימה אחרת. כלומר אופרטור == יחזיר true אך ורק במידה וכל הזוגות של המפתחות והמידע שווים זה לזה לחלוטין ובאותו הסדר ויחזיר false בכל מצב אחר. אופרטור != יחזיר את ההפך (ממשו אותו בעזרת ==)

הנחיות נוספות

- בתרגיל זה באופן ספציפי מותר להשתמש אך ורק במחלקת string מהספרייה הסטנדרטית. כלומר, ה include היחיד מהספרייה הסטנדרטית יהיה בקובץ MyLinkedList.hpp והוא יהיה של string.
- הקוד שלכם צריך להתקמפל בלינוקס אובונטו ע"י ה Makefile שניתן לכם. שימו לב שעליכם להשלים את ה google test שניתן בקובץ MyLinkedListTest.cpp מכיוון שהוא לא בודק את כל הפונקציות ולא בודק את כל סוגי השימוש האפשריים בפונקציות.
- בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות היא אחריותכם. חישבו על מקרי קצה, חלק מהציון ניתן על עמידה בבדיקות אוטומטיות.

- ניתן לממש את הקוד ע"י מחלקה יחידה. ניתן (אך לא חובה) להשתמש במחלקה נוספת (MyLinkedListNode) שתייצג איבר ברשימה. שימו לב כי ניתן וצריך לממש את כל המחלקה MyLinkedListNode בקובץ MyLinkedList.cpp (כך יותר נכון מבחינת Information hiding). בעיקרון רצוי שזאת תהיה מחלקה שרק מכילה מידע ללא לוגיקה מסובכת ואז אין צורך לבדוק אותה (ע"מ לבדוק אותה היינו צריכים שיהיה עוד header).
- בתרגיל זה אתם יכולים להניח שאין שגיאות (הקצאות הזיכרון תמיד מצליחות, הקלט תמיד חוקי וכו').
- חובה להשלים את הגוגל טסט בקובץ MyLinkedListTest.cpp
- ע"מ להחזיר מספרי איברים השתמשו בטיפוס `size_t`
- **העדיפו קוד נקי ויפה על פני קוד "יעיל"**. לדוגמא, כמו שכתבנו ממשו את אופרטור `!=` ע"י שימוש באופרטור `==`. הערת העשרה: לרוב אנשים חוזרים על קוד לשם "יעילות" אך מנגנונים כמו `inline` ואופטימיזציות של הקומפיילר גורמים לכך שהקוד לא באמת יותר יעיל מקוד נקי ומסודר.

הנחיות הגשה

כזכור עליכם להגיש קובץ ששמו מספר ת.ז. שלכם כמו שהיא מופיעה באתר המודל נקודה קוד zip. לדוגמא, אם מספר ת.ז. שלי הוא 12345678 אז שם הקובץ יהיה:

12345678.zip

הקובץ יכיל את הקבצים הבאים:

MyLinkedList.hpp
 MyLinkedList.cpp
 MyLinkedListTest.cpp

מומלץ מאוד להשתמש בקובץ Makefile המצורף לצורך קימפול והרצת התרגיל, יצירת ובדיקת קובץ zip.

בהצלחה!