סדנת תכנות בשפת C++, מס' קורס 67319 - 2018

תרגיל 1

היכרות עם השפה, classes, const , references,
dynamic allocation, operators overloading

תאריך הגשה: 30/08/2018 23:55
הגשה מאוחרת (בהפחתת 10 נקודות): 31/08/2018 23:55
תאריך ההגשה של הבוחן: 30/08/2018 23:55

הנחיות חשובות לכלל התרגילים:

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. עבור כל פונקציה בה אתם משתמשים, עליכם לוודא שאתם מבינים היטב מה הפונקציה עושה גם במקרי קצה (התייחסו לכך בתיעוד). ובפרט עליכם לוודא שהפונקציה הצליחה.
5. בכל התרגילים במידה ויש לכם הארכה, או שאתם מגישים באיחור. **חל איסור להגיש קובץ כלשהו בלינק הרגיל (גם אם לינק overdue טרם נפתח)**. **מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.**
6. אין להגיש קבצים נוספים על אלו שתדרשו. ובפרט אין להגיש קובץ README אלא אם צוין במפורש שיש צורך בכך. (לדוגמא, בתרגיל זה אין צורך להגיש).
7. עליכם לקמפל עם הדגלים Wall -Wextra -Wvla -std=c++14 -g ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה משמעותית בציון התרגיל.** למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex1.cpp יש להריץ את הפקודה:

```
g++ -Wextra -Wall -Wvla -std=c++14 -g ex1.cpp -o ex1
```

8. 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)
9. לאחר ההגשה, בדקו את הפלט המתקבל בקובץ ה-PDF שנוצר מהpresubmission 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.**
10. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. בדקו מקרי קצה.
   במידה וסיפקנו לכם קבצי בדיקה לדוגמא, השימוש בהם יהיה על אחריותכם. **במהלך הבדיקה הקוד שלכם ייבדק מול קלטים נוספים לשם מתן הציון.**

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

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 C (char *).
2. יש להשתמש בספריות סטנדרטיות 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. הקפידו מאוד על שימוש במילה השמורה const בהגדרות המתודות והפרמטרים שהן מקבלות:
המתודות שמקבלות משתנה ייחוס (reference) או מצביע ואינן משנות אותו – 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 לשחרר את כל הזיכרון שאתם מקצים (השתמשו ב-valgrind כדי לבדוק שאין לכם דליפות זיכרון).
7. שימו לב שהאלגוריתמים שלכם צריכים להיות יעילים.
8. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.

הנחיות ספציפיות לתרגיל זה:

1. עליכם לוודא שהקוד שלכם רץ באופן תקין וללא דליפות זכרון. לשם כך עליכם להשתמש בתוכנת valgrind (ראו פירוט בהמשך).
2. חל איסור להשתמש במבני נתונים מוכנים בתרגיל (כדוגמת STL) שימוש כזה יוביל לפסילת הסעיף הרלוונטי.
3. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.
4. שימו לב שאתם מכירים כל פונקציה בה אתם משתמשים ושאתם בודקים עבור כל פונקציה שהיא הצליחה.
5. בהמשך הקורס נלמד לטפל במצבים לא צפויים באמצעות מנגנון החריגות (exceptions). מכיוון שמנגנון זה עוד לא נלמד, בתרגיל זה תוכלו להניח כי התוכנית פועלת כצפוי או לאמץ ערך ברירת מחדל, כפי שמפורט בהמשך.

מספר שלם גדול כרצוננו – Big Integer

בחלק זה של התרגיל זה תבנו מחלקה המחזיקה מספר שלם המוגבל בגודלו אך ורק על ידי זיכרון המחשב, על ידי שמירת הספרות במחרוזת שניתן להגדיל לפי הצורך. כך למשל עבור המספר 1234 ומחרוזת השמורה במשתנה בשם _data, ניתן לגשת לספרה הקטנה ביותר על ידי:

_data[0] // gives '4'

ולקבל את המספר המתאים על ידי:

(_data[0] – '0') // gives the number 4 (as an int)

עליכם לממש עבור המחלקה את הפעולות הבאות:

1. בנאים:
   a. בנאי ברירת מחדל המאתחל את המספר ל-0.
   b. בנאי העתקה המקבל big_integer.
   c. בנאי המקבל מספר שלם רגיל (int).
   d. בנאי המקבל מחרוזת (std::string). לא ניתן להניח כי המחרוזת מכילה ייצוג תקני של מספר. באם הייצוג אינו תקני יש לאתחל את המספר לאפס.
2. אופרטורי חיבור, + ו- =+.
3. אופרטורי חיסור, − ו- =−.
4. אופרטורי כפל, * ו- =*.
5. אופרטורי חילוק בשלמים (ללא שארית), / ו- =/. עבור חילוק באפס יש להחזיר אפס.
6. אופרטור שארית, %. עבור חילוק באפס יש להחזיר אפס.
7. אופרטור השמה.
8. אופרטורי השוואה: את האופרטורים ==, ! =, >, <, <=, >=.
9. אופרטור הדפסה <<.

הערות:

- יש לבצע את הכפל, החילוק והשארית באופן יעיל (מסדר גודל של מספר הספרות. רמז: חילוק ארוך).
- ניתן לממש פונקציות נוספות בתוך הקבצים אותם עליכם להגיש.

קבוצת מספרים set

כעת נרצה לבנות מחלקה המייצגת קבוצה של מספרים גדולים: המבנה הוא קבוצה מתמטית, כלומר שאיבר יכול להופיע בה לכל היותר פעם אחת. למחלקה זו תידרשו לממש את הפונקציות והאופרטורים הבאים:

1. בנאים:
   a. בנאי ברירת מחדל המייצר קבוצה ריקה.
   b. בנאי העתקה המקבל my_set.
2. is_in_set הבודקת אם מספר נמצא בקבוצה, מחזירה true אם הוא נמצא ו-false אחרת.
3. add, המקבלת מספר גדול ומוסיפה אותו לקבוצה אם הוא אינו נמצא בה. פונקציה זו מחזירה ערך בוליאני (bool) המעיד אם האיבר הוסף.
4. remove המסירה איבר מן הקבוצה, ומחזירה ערך בוליאני המעיד על מציאה (והסרה) של האיבר.
5. sum_set המחזירה את סכום המספרים הנמצאים בקבוצה.
6. אופרטור חיסור −, אשר מבצע חיסור בין קבוצות[1]. אופרטור זה מחזיר קבוצה המכילה את האיברים מהקבוצה בצד שמאל של האופרטור שאינם נמצאים בקבוצה מצד ימין של האופרטור.
7. אופרטור איחוד |, אשר מחזיר איחוד של שני קבוצות[2].

---

[1] בדרך כלל מסומן $A \setminus B$.
[2] בדרך כלל מסומן $A \cup B$.

8. אופרטור חיתוך &, אשר מחזיר חיתוך בין שני קבוצות[3].
9. אופרטור הדפסה >> המדפיס את הקבוצה על פי סדר מהקטן לגדול, עם ירידת שורה לאחר כל מספר. אם הקבוצה ריקה תודפס ירידת שורה.

הערות:

- מומלץ לממש את הקבוצה כרשימה מקושרת (כל מימוש יעיל באופן סביר יתקבל).
- ניתן להגדיר פונקציות ומחלקות נוספות בתוך הקבצים אותם עליכם להגיש. מומלץ לממש מחלקות נוספות כמחלקות פנימיות פרטיות על מנת לשמור על מימוש כמוס (encapsulation).
- בונוס: מימוש אשר מדפיס את האיברים בזמן $O(N \log N)$ יקבל 5 נקודות בונוס לציון התרגיל (בין על ידי מיון או על ידי שמירה במבנה נתונים ממוין). אם מימשתם את הבונוס, ציינו זאת (ואיך בחרתם לממש אותו) בתיעוד המחלקה.

הערות כלליות למשימות התכנות:

1. התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות, באמצעות קבצי קלט שונים (תרחישים שונים להרצת התכניות). הפלט של פתרונותיכם יושווה (השוואת טקסט) לפלט של פתרון בית הספר. לכן עליכם להקפיד על פורמט הדפסה מדויק, כדי למנוע שגיאות מיותרות והורדת נקודות.
2. קבצי הבדיקה שסיפקנו לכם מכילים דוגמאות מועטות לבדיקה של חלק קטן מהקוד. עליכם לכתוב בדיקות נוספות על מנת לוודא את תקינות המימוש שלכם. מותר (ואף מומלץ) לשתף את הבדיקות שכתבתם בפורום המיועד לכך, כל עוד הן אינם חושפות פרטי מימוש. בדיקות טובות יזכו את כותביהן בנקודות בונוס.

הגשה:

1. עליכם להגיש קובץ tar בשם ex1.tar המכיל **רק** את הקבצים הבאים:
   - big_integer.h
   - big_integer.cpp
   - my_set.h
   - my_set.cpp
   - קובץ Makefile התומך לפחות בפקודות הבאות:
     - make all – הידור ויצירת שתי תוכניות הבדיקה: testint ו- testset
     - make testint – הידור, יצירה והרצה של תוכנית testint (עם קובץ בשם big_int_tester.cpp שעשוי להיות שונה מזה שסופק לכם).
     - make testset – הידור, יצירה והרצה של תוכנית testset (עם קובץ בשם my_set_tester.cpp שעשוי להיות שונה מזה שסופק לכם).
     - make clean – ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-Makefile (וניתן לשחזר באמצעות קריאה מחודשת לפקודות ה- make המתאימות)

   - extension.pdf - רק במקרה שההגשה היא הגשה מאושרת באיחור בקישור ex1_late (מכיל את האישורים הרלוונטים להארכה).

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

---

[3] בדרך כלל מסומן $A \cap B$.

~plabc/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)

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

~plabcpp/www/ex1/presubmit_ex1

בהצלחה!