# מבוא לתכנות מערכות
# תרגיל בית מספר4 (C++)

## סמסטר אביב 2016

תאריך פרסום: 25.5.2016
תאריך הגשה: 21.6.2016
משקל התרגיל: 10% מהציון הסופי (תקף)
מתרגל אחראי לתרגיל: איציק גולן

תרגיל זה מחולק לשלושה חלקים, כך שניתן להתחיל לפתור אותו לפני שנלמד כל החומר עבור התרגיל.
**חלק ראשון** – מחלקות והעמסת אופרטורים.
**חלק שני** – תבניות.
**חלק שלישי** – ירושה, פולימורפיזם ו- STL.

עבור כל אחד מהחלקים מסופקים לכם קבצי דוגמא. עליכם לדאוג כי התכנית תתקמפל ותעבור את הבדיקות יחד עם קבצי הדוגמא המסופקים.

**אין להשתמש בSTL בחלק הראשון והשני.**

**שימו לב להקפיד על const-correctness לכל אורך המימוש. קוד שיגדיר ארגומנטים או מתודות בצורה שגויה עלול לא להתקמפל.**

**הקפידו להגדיר ארגומנטים וערכי חזרה כReference (או const-reference) במקרים המתאימים. כלל אצבע – תמיד עדיף const-reference היכן שמתאפשר.**

## חלק א' – Apartment

בחלק זה תממשו את המחלקה Apartment מתרגיל 1, בשינויים קלים. **אין להשתמש בmalloc/free, אלא רק בnew/delete.**

אובייקט מסוג Apartment מורכב מלוח משבצות המתאר את מבנה הדירה, וכן ממחיר הדירה. המחיר הינו מספר שלם **אי שלילי** כלשהו (מחיר 0 הוא מחיר חוקי), ומימדי הדירה **חיוביים**. כמו כן, הדירה שומרת את המידות של לוח המשבצות הנתון.

בקובץ Apartment.h המסופק לכם, מופיע enum המגדיר את הטיפוס:

```
Enum SquareType {EMPTY, WALL, NUM_SQUARE_TYPES};
```

א. ממשו בנאי עם החתימה

```
Apartment (SquareType** squares, int length, int width, int price);
```

גם בתרגיל זה, **אין להניח שהתוכן של squares חוקי לאחר הקריאה לבנאי!** במידה ואחד הארגומנטים אינו חוקי, יש לזרוק חריגה מסוג IllegalArgException, המוגדרת בקובץ הheader המסופק.

ב. אם צריך, ממשו הורס, בנאי העתקה, ואופרטור השמה. אם לא, הכריזו עליהם כdefault כפי שנלמד בתרגול.

ג. ממשו מתודה בשם getTotalArea() המחזירה int, המייצג את מספר המשבצות הריקות בדירה.

ד. ממשו מתודה בשם getPrice () המחזירה int, המייצג את מחיר הדירה.

ה. ממשו מתודה בשם getLength() המחזירה int, המייצג את אורך הדירה (מספר השורות).

ו. ממשו מתודה בשם getWidth () המחזירה int, המייצג את רוחב הדירה (מספר העמודות).

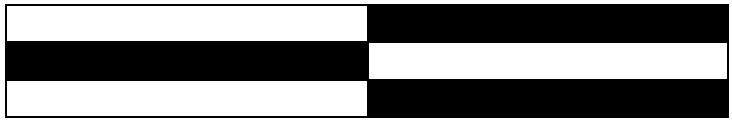
ז. ממשו את operator+=, המחבר שתי דירות יחד ומחליף את תוכן הדירה בצד השמאלי של האופרטור. חיבור שתי דירות יעשה באופן הבא:

a. אם הדירות באותו הרוחב, הדירה המחוברת (השנייה) "תודבק" *מתחת* לדירה אליה מחברים (הראשונה). כלומר, העמודה הj של הדירה החדשה תורכב משרשור של העמודות הj של הדירה הראשונה והשנייה.

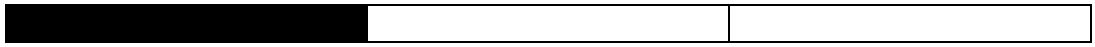
b. אם הדירות באותו האורך, הדירה המחוברת (השנייה) "תודבק" *מימין* לדירה המתחברת (הראשונה). כלומר, השורה הi של הדירה החדשה תורכב משרשור של השורות הi של הדירה הראשונה והשנייה.

c. אחרת, הדירה המחוברת (השנייה) "תודבק" *מתחת* לדירה אליה מחברים (הראשונה). כלומר, העמודה הj של הדירה החדשה תורכב משרשור של העמודות הj של הדירה הראשונה והשנייה. לאחר מכן, המשבצות ה"חסרות" להשלמת מלבן, יתווספו בתור Wall, לדוגמה:

a =



b =



a+b =

ח. ממשו את operator(), המקבל שני פרמטרים מסוג int – הראשון שורה והשני עמודה, ומחזיר רפרנס למשבצת המתאימה. **אל תשכחו לממש את שתי הגרסות – const, non-const**. במידה ואחד הפרמטרים אינו חוקי (מחוץ לגבולות הדירה), יש לזרוק חריגה מסוג OutOfApartmentBoundsException.

ט. ממשו את operator<, הקובע כי דירה "קטנה" מדירה אחרת אם ורק אם היחס בין המחיר לשטח שלה קטן יותר (price/TotalArea). במקרה של שוויון בין היחסים, הדירה עם המחיר הנמוך יותר נחשבת "קטנה" יותר. חשבו היכן לממש אופרטור זה? בתוך המחלקה או מחוץ למחלקה? במידה ובחרתם בפונקציה חיצונית, האם בהכרח צריכים להכריז עליה כfriend. זכרו שעבור אופטורים המקבלים 2 ארגומנטים, נוכל לתמוך בהמרות אוטומטיות (במידה ונרצה להוסיף יכולת כזו למחלקה שלנו) אם נממש את הפונקציה מחוץ למחלקה, ושתמיד עדיף שמה לא **חייב** להיות friend – לא יהיה.

י. ממשו את operator+. גם כאן התחשבו בנקודות המתוארות בסעיף הקודם, וחישבו כיצד ניתן להשתמש בדברים שכבר מימשתם על מנת לפשט מאוד את מימוש הפונקציה.

## חלק שני – תבניות (templates)

בחלק זה נממש מבנה נתונים גנרי מסוג SortedSet, אותו תממשו בקובץ SortedSet.h (שנו את הקובץ הקיים). מבנה הנתונים SortedSet מחזיק אוסף **ממוין** של אובייקטים מטיפוס מסויים. האובייקטים השונים נשמרים כך שאין שני אובייקטים זהים בתוכו.

### מחלקת iterator

על מנת לאפשר מעבר על איברי המבנה, עליכם לממש גם את המחלקה הפנימית iterator. איטרטור יכול "להצביע" שהוכנסו לSet, או להצביע ל"סוף" המבנה – כלומר להימצא במצב שאינו מצביע על אף איבר, לאחר שהשתמשנו בו כדי לעבור על כל האיברים השמורים בSortedSet. לאחר כל פעולה של שינוי הSortedSet, כלומר הוספה או הוספה של איבר, כל האיטרטורים לאותו מבנה שנוצרו לפני פעולה זו הופכים להיות invalid, ופעולה עליהם לא מוגדרת (ולכן גם לא ייבדקו פעולות על איטרטורים invalid). האיטרטור צריך לתמוך בפעולות הבאות:

- קידום: ע"י אופרטור++. שימו לב כי עליכם לממש את שתי הגרסאות של האופרטור. חישבו מהו ערך ההחזרה המתאים עבור כל גרסה של האופרטור. במידה והאיטרטור מצביע לסוף המבנה, הפעלת האופרטור אינה מוגדרת.
- Dereferencing ע"י האופרטור האונארי*. החזרת האיבר עליו מצביע האיטרטור. במידה והאיטרטור מצביע לסוף המבנה, הפעולה אינה מוגדרת (ולא תיבדק). עם זאת, כאשר אתם משתמשים באיטרטור, **באחריותכם** לוודא שאינכם מבצעים dereference לאיטרטור שמצביע על סוף המבנה (למשל ע"י השוואה לאיטרטור המצביע לסוף המבנה). אין לאפשר למשתמש בפעולה זו אפשרות לשנות את האובייקט המוחזר, כיוון שמיקומו היחסי של האובייקט בתוך המבנה עלול להשתנות כתוצאה מפעולות עליו.
- השוואה: אופרטור == ואופרטור =!. שני איטרטורים שווים אם הם מצביעים לאותו איבר באותו אובייקט SortedSet, או אם שניהם מצביעים לסוף של אותה ה- SortedSet, ושונים אחרת. השוואת איטרטורים שמקורם מSortedSet שונים היא פעולה לא מוגדרת, ולא תיבדק.
- בנוסף, על האיטרטור לתמוך בהעתקות ובהשמות של עצמו.

## מחלקת SortedSet

המבנה צריך לתמוך בפעולות הבאות:

- איטרציה על ה- map תתבצע באמצעות יחס סדר על המפתחות המוגדר לפי מחלקה גנרית (Compare בקובץ הנתון), אשר מממשת אופרטור סוגריים עגולים בינארי, המקבל אובייקטים מהטיפוס הנשמר בSortedSet ומחזיר bool. אם compare(element1, element2) מחזיר true, אז נניח כי element1<element2 (ולכן element1 יופיע בשלב מוקדם יותר באיטרציה).
- ניתן להניח כי שני הטיפוסי T מנהל את הזיכרון של עצמם באופן תקין.
- ניתן להניח כי שני הטיפוסים T מממשים copy constructor, destructor ואופרטור השמה. **שימו לב! הקפידו כי אינכם מניחים אף הקלה מקלה נוספת על הטיפוסים הגנריים. קוד שאינו עומד בדרישה זו עשוי לא להתקמפל ולגרור ציון 0 בבדיקה הרטובה!**
- החזרת איטרטור לתחילת ה-SortedSet בעזרת הפונקציה begin.
- החזרת איטרטור לסוף ה- SortedSet בעזרת הפונקציה end. כפי שנכתב קודם, איטרטור זה לא מצביע לאיבר חוקי.
- חיפוש איבר בעזרת הפונקציה find, המקבלת פרמטר אחד מטיפוס T, ומחזירה איטרטור לאיבר המתאים במבנה אם קיים כזה, או איטרטור לסוף המבנה במידה והאיבר לא קיים במבנה.
- הוספת איבר ל- SortedSet ע"י הפונקציה insert. הפונקציה תחזיר true במידה והאיבר אינו היה קיים קודם לכן, ו-false אחרת. במקרה שהאיבר כבר היה קיים, המבנה לא ישתנה.
- הסרת איבר מה- SortedSet ע"י remove. הפונקציה מקבלת אובייקט מטיפוס T בלבד. במידה והאיבר המתאים לא נמצא, על הפונקציה להחזיר false, ואחרת true.
- החזרת מספר האיברים ע"י size.
- חיתוך של שני SortedSet ע"י &operator המקבל שני SortedSet כארגומנטים. יוחזר אובייקט SortedSet חדש, המכיל את האיברים הנמצאים בשני המבנים.
- איחוד של שני SortedSet ע"י |operator המקבל שני SortedSet כארגומנטים. יוחזר אובייקט SortedSet חדש, המכיל את האיברים הנמצאים בלפחות אחד משני המבנים.
- הפרש של שני SortedSet ע"י -operator המקבל שני SortedSet כארגומנטים. יוחזר אובייקט SortedSet חדש, המכיל את האיברים הנמצאים במבנה הראשון (השמאלי), אבל לא נמצאים במבנה השני.
- הפרש סימטרי של שני SortedSet ע"י ^operator המקבל שני SortedSet כארגומנטים. יוחזר אובייקט SortedSet חדש, המכיל את האיברים הנמצאים באחד משני המבנים, אבל לא נמצאים במבנה השני.
- בנוסף, על ה- SortedSet לתמוך בהעתקות והשמות של המחלקה.

## חלק שלישי – ירושה, פולימורפיזם, וSTL

**בחלק זה הנכם רשאים להשתמש בmap, list, string, stringstream מSTL בלבד. כמו כן, הינכם רשאים (וצריכים) להשתמש בSortedSet שכתבתם בסעיף הקודם.**

בחלק זה נממש מערכת הנקראת Publisher-Subscriber. מערכת זו הינה מערכת סקלאבילית (ניתנת להרחבה באופן פשוט) להעברת הודעות. במערכת קיימות שלוש יישויות מרכזיות – Publisher, Subscriber, Broker.

הPublisher מפרסם לBroker את נושא (או נושאי) ההודעות שהוא מסוגל לשדר, הנקרא Topic. כמו כן, הSubscriber נרשם לנושא (או נושאים) של ההודעות המעניינות אותו אצל הBroker). כאשר לPublisher יש הודעה שהוא רוצה לפרסם, הוא מעביר אותה, יחד עם הTopic המתאים, אל הBroker. לבסוף, הBroker מעביר את ההודעה לכל הSubscribers הרלוונטים.

ארכיטקטורה זו מאפשרת הפרדה בין הרכיבים השונים במערכת, כך שמי שמשדר את ההודעות לא צריך להכיר את כל מי שרוצה לקבל אותן.

*ניתן להניח כי כל Broker נשאר חי מרגע יצירתו ועד סוף התוכנית, וכי Client נשאר חי מהרגע שהוא מבצע publish/subscribe ועד לסוף התכניות.*

### Broker

מסופק לכם קובץ בשם BrokerIfc.h, המכיל הגדרה של מחלקה אבסטרקטית. **אין לשנות קובץ זה, ואין להגיש אותו.** עליכם לממש את מחלקת Broker היורשת מBrokerIfc, וממשת את הפונקציות האבסטרקטיות המוגדרות בה - רישום/הסרה של Topic לשידור, רישום/הסרה של Topic לקבלה, ושליחת הודעה. הפונקציות של המחלקה BrokerIfc אמורות להיקרא רק מתוך מהמימוש של הPublisher וה-Subscriber, לכן מחלקות אלו הוכרזו כ"חברות", והמתודות לעיל כprivate. **הקפידו** לשמור על הvisibility הזה (private) גם במחלקה Broker היורשת.

בנוסף, עליכם להכריז ולממש שתי פונקציות *פומביות* נוספות – sendMaintenanceMessageAny, ו-sendMaintenanceMessageAll. שתי הפונקציות האלה מקבלות std::list<Topic> בארגומנט הראשון, ו-std::string בארגומנט השני. הפונקציה הראשונה שולחת הודעה בקרה מהBroker לכל הPublishers/Subscribers שרשומים/פרסמו לTopic *כלשהו* מבין הTopics שברשימה. הפונקציה השנייה שולחת רק לאלה שרשומים/פרסמו *לכל* הTopics שברשימה. פונקציות אלה שולחות את ההודעות בצורה לא מוצפנת (יפורט בהמשך), ללא קשר לטיפוס האמיתי של האובייקטים המקבלים אותה.

### Client

מחלקת בסיס לPublisher/Subscriber.

- תכיל בנאי המקבל priority, &Broker, ו-&std::ostream. (priority ראשון וכו'). העדיפות משמשת לקביעה של סדר קבלת ההודעות מהBroker (בין אם הודעות רגילות או הודעות בקרה). מספר נמוך יותר יקבל הודעות קודם. במקרה של שוויון בעדיפות, הClient עם הId הנמוך יותר יקבל הודעות קודם. במקרה של priority<0, יש לזרוק חריגה IllegalPriority, המוגדרת בקובץ הממשק המסופק. הארגומנט האחרון יוגדר בברירת המחדל כ-std::cout, ונתייחס אליו בהמשך התרגיל כ-messagesSink. **כל ההודעות (בקרה ורגילות) המגיעות לClient או למחלקה יורשת שלו, יופנו אל הostream& הנתון כאשר הן מתקבלות אצל האוביקט, כאשר לסוף כל הודעה ישורשר גם std::endl.**
- במקרה הצורך – הגדירו בנאי העתקה ואופרטור השמה. אם אתם מעוניינים להשתמש בברירת המחדל – ציינו כך במפורש כפי שנלמד בתרגול.
- getPriority המחזירה int שהוא הpriority שסופק ביצירת האובייקט.

- getId. מחזיר int שהוא מספר מזהה **ייחודי** לClient. חשבו איך ניתן לממש דבר כזה בעזרת משתנה סטטי של המחלקה.
- receiveMaintenanceMessage. פונקציה זו מקבלת std::string, שהוא הודעת הבקרה שנשלחה מהBroker, ומפנה string בפורמט הבא אל הmessagesSink :

  "Client #<client_id> received maintenance message: <message_content>"

  למשל:

  "Client #12 received maintenance message: This is a test message"

## Subscriber

- תכיל בנאי זהה לזה של Client (כולל ערך ברירת המחדל).
- במקרה הצורך – הגדירו בנאי העתקה ואופרטור השמה. אם אתם מעוניינים להשתמש בברירת המחדל – ציינו כך במפורש כפי שנלמד בתרגול.
- subscribeToTopic, מקבלת Topic ולא מחזירה דבר. לאחר הפעלת מתודה זו, כל הודעה שנשלחת ע"י Publisher אל הTopic הנתון, תגיע את הSubscriber (כלומר תופעל מתודת receiveMessage שתתואר למטה, עם ההודעה).
- unsubscribeToTopic, מקבלת Topic ולא מחזירה דבר. לאחר הפעלת מתודה זו, הודעות שנשלחות לTopic הנתון מPublisher לא יגיעו אל הSubscriber.
- unsubscribeAll – אינה מקבלת או מחזירה ארגומנטים. לאחר הפעלת המתודה לא תתקבלנה שום הודעה ל Subscriber מאף Publisher.
- receiveMessage, פונקציה וירטואלית המקבלת string, Topic, ו-Client (string ראשון וכו') ולא מחזירה דבר. במידה וההודעה הגיעה עם Topic שעליו לא התבצע subscription, תיזרק חריגה מסוג NonSubscribedTopic המוגדרת בSubscriber.h. אחרת, המתודה תפנה string בפורמט הבא אל ה messagesSink:

  “Topic: <topic>. Sender: #<sender_id>. Receiver: #<receiver_id>. Message: <message_content>”

  למשל:

  “Topic: Cat Videos. Sender: #0. Receiver: #2. Message: This is a cat video”

## Publisher

- תכיל בנאי זהה לזה של Client (כולל ערך ברירת המחדל).
- במקרה הצורך – הגדירו בנאי העתקה ואופרטור השמה. אם אתם מעוניינים להשתמש בברירת המחדל – ציינו כך במפורש כפי שנלמד בתרגול.
- publishTopic, מקבלת Topic ולא מחזירה דבר. לאחר הפעלת מתודה זו, כל הודעה שנשלחת ע"י Publisher זה אל הTopic הנתון, תגיע את הSubscribers הרשומים אליו.
- unpublishTopic, מקבלת Topic ולא מחזירה דבר. לאחר הפעלת מתודה זו, הPublisher **לא יוכל** לשלוח הודעות לTopic.
- unpublishAll – אינה מקבלת או מחזירה ארגומנטים. לאחר הפעלת המתודה הPublisher **לא יוכל** לשלוח הודעות לאף Topic.
- sendMessage, פונקציה וירטואלית המקבלת string, ו- Topic (string ראשון וכו'). במידה וההודעה נשלחת עם Topic שעליו לא התבצע publishTopic, תיזרק חריגה מסוג NonPublishedTopic המוגדרת בSubscriber.h. אחרת, ההודעה תועבר לSubscribers הרלוונטים דרך הbroker. המתודה אינה מחזירה דבר.

## EncryptionSubscriber, EncryptionPublisher

נרצה לממש מחלקות היורשות מSubscriber/Publisher, אשר מסוגלות לשלוח ולקבל הודעות מוצפנות. ההצפנה שבה נשתמש תהיה פשוטה מאוד – כל תו (char) במחרוזת, יעבור XOR עם מפתח כלשהו שיתקבל בבנאי. XOR נוסף עם אותו המפתח כמובן יחזיר את כל התווים למצבם המקורי.

- המחלקות יכילו בנאי המקבל 4 פרמטרים – שלושה כמו של Client, ופרמטר רביעי מסוג char שיהווה מפתח הצפנה. כיוון שפרמטר עם ערך ברירת מחדל חייב להופיע אחרון, הפרמטר השלישי יהיה הkey, והרביעי יהיה ה&ostream עם ערך ברירת המחדל std::cout.
- במקרה הצורך – הגדירו בנאי העתקה ואופרטור השמה. אם אתם מעוניינים להשתמש בברירת המחדל – ציינו כך במפורש כפי שנלמד בתרגול.
- המחלקות יבצעו override למתודה הוירטואלית sendMessage/receiveMessage על מנת לשלוח/לקבל הודעה מוצפנת. פעולת XOR מיוצגת בC++ בעזרת האופרטור '^'. כמובן המתודה receiveMessage צריכה גם היא לכתוב לתוך הmessagesSink בפורמט המתאים (קיימת דרך פשוטה לבצע זאת ע"י שימוש במימוש של מחלקת האב).

## הנחיות כלליות

1. התעלמו מחריגות מסוג std::bad_alloc
2. יש לשמור על הסתרה ומיעוט שכפולי קוד כנלמד בקורס.
3. יש להקפיד על const correctness.
4. יש להשתמש בהעמסת אופרטורים כאשר הדבר מתאים.
5. המימוש חייב לציית לכללי כתיבת הקוד המופיעים באתר תחת Course Material, Code Conventions. אי עמידה בכללים אלו תגרור הורדת נקודות. **שימו לב שהאורך המקסימלי המותר לפונקציה הוקטן בתרגיל זה!**
6. המערכת צריכה לעבוד על Stud/T2.
7. שימו לב! העדכונים ב- FAQ מחייבים
8. על מנת לעבוד עם C++11 יש להתקין גרסה חדשה יותר של G++. הוראות מפורטות על הורדת הקומפיילר ויצירת פרוייקט עבור C++11 באקליפס:

### How to set and verify the new g++ version for the assignment?

In order to use gcc/g++/gfortran 4.7.1, you should run first:

source /usr/local/gcc4.7/setup.csh # for CSH and TCSH
or:
. /usr/local/gcc4.7/setup.sh # for BASH and SH

verify version by:

gcc --version
gcc (GCC) 4.7.1

Please note that this change will expire after you log out.
Refer to tutorial #2 to see how to make this permanent.

### How do I create get C++11 to work in Eclipse?

Go to Project | Properties | C/C++ Build. Choose C++ Compiler, then Dialect then Language standart and change it to ISO C++11

If your eclipse version is not the newest, you might not have this option.
In this case,
Add the -std=c++11 flag by going into Project | Properties | C/C++ Build. Choose C++ Compiler, Settings and add under miscellaneous "-std=c++11".
In addition, to make Eclipse understand C++11, go to Discovery Options (Also under C/C++ Build). Make sure "C++ Compiler" is chosen and under "Compiler invocation arguments" add "-std=c++11" in the end of this line (with a space before ti of course).

## בדיקות יחידה

עליכם לכתוב בדיקות יחידה עבור כל אחת מהמחלקות אותן תכתבו בחלק הרטוב. את בדיקות היחידה יש לשים בתוך תיקייה בשם test. יש להגיש קבצים אלו ללא פונקציית main.

מספר דגשים:

- הקפידו לבדוק גם את התנהגות הקוד במצבי שגיאות, ולא רק עבור פרמטרים וקלט מוצלח.
- הקפידו על איכות הקוד גם בבדיקות – הבדיקות צריכות להיות קצרות, מחולקות לפונקציות וקריאות, כך שבמקרה כישלון יהיה קל למצוא את הסיבה לבאג.
  - מומלץ לחלק את הבדיקות כך שלכל פונקציה בכל מחלקה קיימת פונקציה הבודקת אותה (שימו לב כי הקבצי Example אינם קבצי בדיקות ואינם מכסים את כל המקרים, מטרתם היא לוודא הבנת התרגיל)
- השתמשו בקבצים mtmtest.h ו-mtmtest.cpp כדי לבדוק את הקוד שלכם.

**המלצה:** כתיבת בדיקות לוקחת זמן אך מקלה משמעותית על תחזוקת קוד בהמשך ועל צמצום הזמן הדרוש לדיבוג התכנית. מומלץ לכתוב את הבדיקות בד בבד עם הקוד. למשל לפני שאתם כותבים פונקציה מסוימת, כתבו את הפונקציה הבודקת אותה. את כתיבת הבדיקות כדאי לעשות כאשר חושבים כיצד ניתן "להכשיל" את הקוד ובכך לוודא את איכות הקוד במקרה קצה. כתיבת כל הבדיקות לאחר סיום התרגיל תהיה מעיקה משמעותית ותתבצע אחרי כל השלבים שבהן הבדיקה יכלה להועיל בפועל.

**המלצה 2:** אל תתחילו לדבג לפני שיש לכם בדיקות יחידה.

## הידור, קישור ובדיקה

השתמשו בפקודת הקומפילציה הבאה:

```
>g++ -std=c++11 -Wall -Werror -pedantic-errors -DNDEBUG *.cpp \
-o [program name] [source code files]
```

התרגיל ייבדק בדיקה יבשה ובדיקה רטובה.
הבדיקה היבשה כוללת מעבר על הקוד ובודקת את איכות הקוד ועמידה בדרישות במימוש.
הבדיקה הרטובה תהיה אוטומטית ותכלול בדיקות אוטומטיות לוידוא נכונות הקוד.

**עליכם לבדוק שהקוד שלכם פועל כצפוי! בדיקת הקוד מהווה חלק מהתרגיל! תרגיל אשר אינו עובר בדיקות יקבל 0 בבדיקה הרטובה, לא יהיו הנחות בנושא זה.**

- וודאו את נכונות התכנית שלכם **במקרים כלליים ומקרי קצה**.
- מומלץ לחשוב על הבדיקות **כבר בזמן כתיבת הקוד**.
- נסו ליצור **מס' רב של בדיקות קצרות** ולא בדיקה אחת גדולה שכישלון בה אינו מסגיר את התקלה.
- ודאו את נכונות הקוד לפני ההגשה **וגם לאחר שינויים קטנים** ולכאורה לא משמעותיים!
- **שימו לב לדליפות זיכרון** – הן ייבדקו בצורה אוטומטית! אפשר להשתמש ב-valgrind כדי לוודא שאין לכם דליפות זיכרון. Valgrind הוא כלי המותקן ב-T2 ומיועד, בין היתר, למציאת דליפות זיכרון.

**שימו לב: כל קבצי הקוד צריכים להימצא בתיקיית הבסיס שלקובץ ה-zip המוגש!**

## קבצים שסופקו על ידינו

לאורך התרגיל הרטוב מופיעות התייחסויות לקבצים שמסופקים עם התרגיל. את כל הקבצים ניתן למצוא בספריה. הקפידו לא לשנות את שמות המתודות והטיפוסים המופיעים בקבצים אלו.

~mtm/public/1516b/ex4

## הגשה

יש להגיש קובץ zip בעל הפורמט הבא:

- תיקיית Tests שבה נמצאים הטסטים אשר כתבתם.
- כל קבצי הקוד עבור התרגיל (h ו- cpp) בתיקייה הראשית של ה- zip (כלומר, לא בתוך אף תיקייה).

אין צורך להגיש קבצים שסופקו על ידנו ולא השתנו.

**כמו כן, אין להגיש את הקובץ main.cpp!** או כל קובץ אחר בו מופיעה הפונקציה main.

## שינויים עדכונים והודעות בנוגע לתרגיל

כל ההודעות הנוגעות בתרגיל ימצאו באתר של הקורס http://webcourse.cs.technion.ac.il/234122 בעמוד ה-FAQ. דף זה יכיל הבהרות וכן שאלות ותשובות נפוצות. רק הודעות דחופות תשלחנה בדואר. עליכם לעקוב אחר האתר והעדכונים שיפורסמו בו.