



פרויקט - חלק 3 - טסטים

תיאור המטלה

בנוסף לפרויקט, הנכם נדרשים להכין סט של 10 טסטים שבדקים אספקטים מסוימים בקומפיילר. הטסטים יבדקו את נכונות הקומפיילר שאתם בונים. המטרה היא להתנסות עם שפת התכנות לפני פיתוח הקומפיילר, ובנוסף שיהיה לכם עם מה לבדוק את הפרויקט שלכם. אנחנו ממליצים לכם לבנות את הטסטים כך שהם יבדקו מגוון גדול ככל האפשר של יכולות הנתמכות בשפה ואף קטעי קוד שאינם מתקמפלים עבור השפה הנתונה. המטרה היא לעזור לכם לבדוק את התרגיל המסכם שלכם לפני שליחתו. שימו לב כי בסיום הגשת תרגיל זה אנחנו **נשתף בין כלל הסטודנטים** את הטסטים שתגישו, כך שיווצר לכם מאגר שלם של טסטים לבדיקת התרגיל.

מבנה של טסט בודד

- כל טסט יהיה מורכב מלפחות 4 קבצים: קבצי הקוד, קובץ קלט וקובץ פלט (באתר הקורס מצורפת דוגמה).
 - **קבצי הקוד:** יכיל לפחות קובץ אחד עם פונקציית main.
 - **קובץ הקלט:** יכיל ערכים לפקודות ה-read אם קיימות. ערך ראשון בשורה ראשונה של הקובץ, ערך שני בשורה שניה של הקובץ, וכו'.
 - **קובץ הפלט:** יכיל את הפלט של פקודות ה-write ושל הסימולטור.
 - **קובץ תוצאה:** קובץ ריק בשם pass/fail. אם קיים קובץ pass נצפה שהטסט יעבור בהצלחה, אם קיים קובץ fail נצפה שהטסט ייפול על שגיאת קומפיציה (אין בדיקה של הודעת השגיאה).
- הערה: במקרה ואין בטסט צורך בקובץ קלט ופלט, יש לצרף בכל מקרה את הקבצים input ו-output ריקים.

דוגמה

הדוגמה המתוארת תופיע גם באתר הקורס לעיונכם. נבנה טסט שבדק פעולות אריתמטיות של מספרים שלמים.

קובץ הטסט:

```
int test (a,b,c:int) {
    return ( a + b ) + b * c + (b + 10) / 2;
}

void main() {
    a,b,c,res:int;
    read(a);
    read(b);
    read(c);
    res = test(a,b,c);
    write(res);
}
```

קובץ הקלט:

```
1
2
3
```

הסבר: הערך בשורה הראשונה יהיה לפי סדר פקודות ה-read. כלומר בסופו של דבר a=1, b=2, c=3.

**קובץ הפלט:**

```
Input integer?:Input integer?:Input integer?:15 Reached Halt.
```

הסבר:

כל פקודת read שהסימולטור (rx-vm) מריץ גורמת להדפסה. קיימות שתי פקודות read אפשריות:

READI – generates "Input integer?:"

READF – generates "Input real?:"

תוצאת החישוב היא 15:

$$\begin{aligned}(a + b) + b \cdot c + (b + 10) / 2 &= \\ (1 + 2) + 2 \cdot 3 + (2 + 10) / 2 &= \\ 3 + 2 \cdot 3 + 12 / 2 &= \\ 3 + 6 + 6 &= \\ 15\end{aligned}$$

בסיום כל תוכנית, הסימולטור מדפיס: "Reached Halt." ושורה ריקה לאחר מכן.

דוגמת שגיאה

הדוגמה המתוארת תופיע גם באתר הקורס לעיונכם.

נבנה טסט שנכשל תחבירית.

קובץ הטסט:

```
void main() {
    a:int;
    read(a+"a");
}
```

הסבר:

תוכנית זו תיכשל עם הודעה של שגיאה תחבירית על שורה 3:

"Syntax error: '+' in line number 3"

נזכור כי עלינו להוסיף קובץ ריק בשם fail לתיקיית הטסט כך שהבודק האוטומטי יידע שזוהי מטרת הטסט.

כפי שכבר הסברנו, אין בדיקה של הודעת השגיאה בתרגיל זה (חלק 3 – טסטים).

מבנה ההגשה

יש להגיש קובץ ארכיב מסוג Bzipped2-TAR בשם מהצורה (שרשור מספרי ת.ז – 9 ספרות):

proj-part3-tests-<student1_id>-<student2_id>.tar.bz2

בארכיב יש לכלול 10 תיקיות (תיקיה לכל טסט) באופן הבא:

```
-- test1:
- test.cmm
- input.in
- output.out
- pass/fail
-- test2:
- test.cmm
- another.cmm
- input.in
- output.out
```



```

- pass/fail
.
.
.
-- test10:
- test.cmm
- input.in
- output.out
- pass/fail

```

במילים, כל תיקיה תכיל 4 קבצים לכל היותר לפי הצורך:

- קבצי הקוד: קבצים שנגמרים בסיימת cmm
- קובץ שיכיל את הקלט לטסט: input.in
- קובץ שיכיל את הפלט לטסט: output.out
- קובץ ריק להצהרת תוצאת הטסט: pass/fail

שימו לב: קובץ הארכיב צריך להיות "שטוח" (כלומר, שלא ייצור ספריות משנה בעת הפתיחה אלא הקבצים ייווצרו בספריה הנוכחית). ניתן להשתמש בפקודה מסוג זה:

```
tar cjf proj-part3-tests-012345678-123456789.tar.bz2 ./*
```

אופן הבדיקה

מסופק לכם קובץ הרצה בשם checker וקובץ בשם rx-runtime.rsk שמיועד ללינקר. checker בודק את נכונות הטסט שכתבתם, חשוב ש-rx-runtime.rsk יהיה באותה תיקייה איתו בעת ההרצה. נריץ אותו כך:

```
./checker test.cmm [more_cmm_files.cmm] input.in output.out
(don't forget to adjust relative path of the files!)
```

הקובץ מקמפל את קוד הטסט, מריץ את קובץ ההרצה כאשר קובץ הקלט נכנס כקלט, ולבסוף מתבצע השוואת diff בין הפלט לקובץ הפלט שסופק.

לכלי יש שלושה פלטים אפשריים:

1. True – טסט עבר בהצלחה.
2. False – פלט הטסט לא תואם את פלט הריצה.
3. Failed – שגיאת קומפילציה.

בבדיקה האוטומטית נשתמש בכלי זה כדי לבדוק שאכן כל הטסטים שלכם מקבלים פלט תקין. תרגיל עובר הוא תרגיל אשר **כל 10 הטסטים שלו תקינים** (טסטים שאמורים לעבור צריכים לקבל true וטסטים שאמורים להיכשל אמורים לקבל failed).

הערה: אין להגיש מעל ל-10 טסטים.

הוראות הגשה

- מועד אחרון להגשה: יום ה' 01/01/2026.
- שימו-לב למדיניות בנוגע לאיחורים בהגשה המפורסמת באתר הקורס. במקרה של נסיבות המצדיקות איחור, יש לפנות מראש לצוות הקורס לתיאום דחיית מועד ההגשה.
- ההגשה בזוגות. הגשה בבודדים תתקבל רק באישור מראש מצוות הקורס.
- יש להגיש בצורה מקוונת באמצעות אתר ה-Moodle של הקורס, מחשבונו של אחד הסטודנטים. הקפידו לוודא כי העלאתם את הגרסה של ההגשה אותה התכוונתם להגיש. לא יתקבלו טענות על אי



התאמה בין הקובץ שנמצא ב-Moodle לבין הגרסה ש"התכוונתם" להגיש ולא יתקבלו הגשות מאוחרות במקרים כאלו.

בהצלחה!