מבחן סופי - קורס בדיקות תוכנה 67778 – סמסטר ב' 2019

מועד א' (תאריך: 8 יולי 2019)

מרצים: מיכאל שטאל, שמואל גרשון

אינטל, ירושלים

ורסיה זו כוללת את ההערות שאמרנו בכיתה בזמן המבחן

הוראות:

בבחינה 10 שאלות. כל השאלות יבדקו. הניקוד לכל שאלה מופיע בסוגריים מרובעות ליד מספר השאלה.

כתבו את התשובות במחברת הבחינה. אין צורך להחזיר את דפי הבחינה.

זמן הבחינה – 180 דקות.

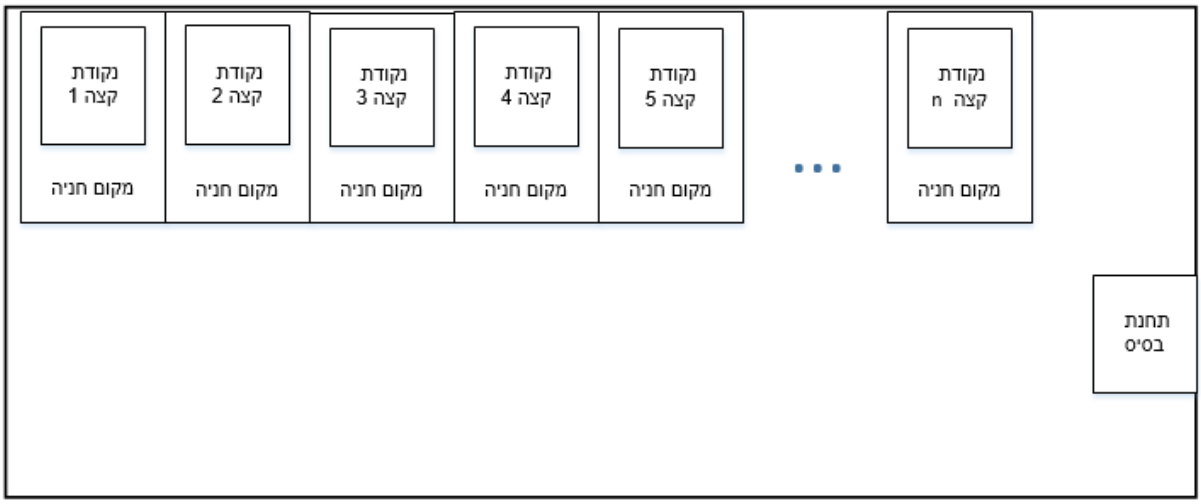
מותר להשתמש בכל חומר עזר מלבד טלפונים, מחשבים או ציוד תקשורת אחר (גם לא דגלי סמפור).

ניתן לרשום הערות על הבחינה או על שאלות ספציפיות, במחברת הבחינה.

בהצלחה!

תרחיש

חניון רכב תת קרקעי בנוי על פי המתואר בשרטוט:



בחניון מותקנת מערכת לזיהוי מקומות חניה פנויים הבנויה בדרך הבאה: בכל מקום חניה מותקנת "נקודת קצה": מערכת אלקטרונית הכוללת רגש (sensor) ומשדר\מקלט. הרגש מזהה אם מקום החניה תפוס או לא. המשדר משדר את מצב מקום החניה פעם אחת (תפוס או פנוי ואת המספר של מקום החניה), וממתין לאישור מתחנת הבסיס שההודעה התקבלה.

על מנת לחסוך באנרגיה, כל משדר עובד בהספק נמוך מאוד, כך שהשידור מגיע רק למערכת האלקטרונית של נקודות הקצה הסמוכות לנקודה המשדרת. המשדר של אותן נקודות קצה חוזר על השידור, ללא כל שינוי, שנייה אחת לאחר שקיבל אותו. בצורה כזו המידע על מצב החניות מועבר מנקודת קצה אחת לשניה, עד שתחנת הבסיס בקיר החניון קולטת את השדר מנקודת הקצה הסמוכה לה ומסמנת את מצב מקום החניה (תפוס; פנוי) בבסיס נתונים.

מיד כשהמידע מגיע, תחנת הבסיס שולחת תשדורת "אישור" בהספק גבוה (שמגיע מיד לכל נקודות הקצה) עם מספר מקום החניה ועם מצב החניה. נקודת הקצה שהמספר שלה הופיע בשדר מוודאת שמצב החניה באישור תואם את מצב החניה בפועל. אם כן, נקודת הקצה לא משדרת שוב ולא מחכה לאישור נוסף. אם מצב החניה באישור ובשטח אינם תואמים – נקודת הקצה מתעלמת מהאישור ומחכה לאישור מתאים.

בכל פעם שמצב החניה משתנה, נקודת הקצה משדרת הודעה חדשה.

אם לא הגיע אישור להודעה שנשלחה מנקודת קצה תוך 15 שניות, הנקודה משדרת שוב הודעה על מצב החניה בפועל.

תחנת קצה משדרת רק לתחנה עם מספר גבוה יותר

התשדורות לא מפריעות אחת לשניה.

המערכת כולה משובצת מחשב. התבקשתם לבדוק את התוכנה של המערכת.

הנחות: כל הציוד הפיזי עובד כראוי (רגשים; משדרים; מקלטים).

דף זה הושאר ריק בכוונה

1) [20]

תארו את פעולת המערכת הנמצאת בנקודות הקצה על ידי טבלת מעברים (state transition table).

טיפ: יתכן שיהיה לכם קל יותר לצייר על טיוטה את תרשים המצבים (state diagram) – ורק אז למלא טבלה.
זכרו שבטבלת מעברים יש לציין גם את האירוע שקרה במערכת וגם את התגובה לאירוע (אירוע X \ פעולה Y). במקרה שאין ביצוע פעולה, ציינו "NULL" במקום Y.

2) [5]
הציעו ארבע בדיקות שאפשר לסווגן כבדיקות מקרי קצה עבור המערכת.

3) [5]
תארו מקרה בדיקה שבעזרתו תבדקו את הדרישה הבאה שהופיע בתיאור המערכת: "אם מצב החניה באישור ובשטח אינם תואמים – נקודת הקצה מתעלמת מהאישור ומחכה לאישור מתאים."
(לא לכתוב מקרה בדיקה שלם. רק את שיטת הבדיקה (Methodology))

במילה "Methodology" אנחנו לא מכוונים למושג ספציפי. הכוונה היא: "הסבירו איך הבדיקה תתבצע".

4) [10]
בדיקת הדרישה לשידור חוזר כל 15 שניות יכולה להעשות בצורה קלה על ידי הגדרת יכולת-עזר לבדיקה (testability hook) שתאפשר בדיקה של פונקציית השידור החוזר גם על מערכת תקינה.
הציעו שתי יכולות עזר (שונות) שניתן לממש לצורך מטרה זו.

5) [15]
לאחר בדיקת המערכת, מצאתם שכשתחנת קצה מקבלת תשדורת מתחנת קצה אחרת על מנת להעביר אותה הלאה, היא משנה את מצב החניה שבתשדורת (אם המסר היה שמקום החניה פנוי, המסר שמועבר הלאה הוא שמקום החניה תפוס; אם המסר היה שמקום החניה תפוס, המסר שמועבר הלאה הוא שמקום החניה ריק). כיתבו דו"ח (bug report) המדווח על פגם זה. על הדו"ח לכלול גם את רמת החומרה של הבאג והסבר למה לדעתכם זו רמת החומרה המתאימה.

6) [10]
הציעו מקרה בדיקה של בדיקת מאמץ (stress) של המערכת. (לא לכתוב מקרה בדיקה שלם. רק מטרה ושיטת בדיקה (Objective and Methodology)

7) [5]

בשל הצורך לעדכן מקומות חנייה באופן ידני (למשל, כדי שיהיה אפשרי לשריין מקומות חנייה), מוסיפים לתחנת הבסיס עמדה בה ניתן להזין מצב פנוי\תפוס. להלן חלק מהקוד. לאיזה סוג של תקיפת הבטחה המערכת פגיעה? תאר בצורה פשטנית את ההתקפה.

Compare_bytes returns 0 when str != '…' (when the strings are not the same)

```
/* Sanitize data to make sure it's safe before modifying
database*/
void SetStatus(int slot, char * str)
{
      // Allocate enough space to write 'free' or 'used'
      char buffer[8];

      if(slot >=1 && slot <= MAX_SLOT_NUMBER)
      {
            if(!compare_bytes(4, *str, 'Free') and
               !compare_bytes(4, *str, 'Used'))
            {
                  strcpy(buffer,str);
                  // Update the database
                  UpdateDatabase(slot, *buffer);
                  return(0)
            }
      }
      return(-1)
}
```

8) [5]

כתבו לפחות שתי דוגמאות לבדיקות שיווצרו עבור fuzzing של הפונקציה הנ"ל אם נשתמש ב:

א) Dumb Fuzzing
ב) Smart Fuzzing

(שתי דוגמאות לכל שיטה)

סוף התרחיש

9) [15] [5 לכל סעיף]

הערה: קיראו את כל סעיפי השאלה לפני שמתחילים לענות עליה.

א) כתבו מקרה בדיקה המדמה מקרה שימוש, עבור אפליקציית WhatsApp. השתמשו במבנה הבא:

Setup and Pre-conditions

1.
2.
3.
...

Steps	Expected Results

ב) קבוצת הפיתוח של WhatsApp החליטה לפתח תשתית לאוטומצית בדיקות מסוג Keyword Driven Test (KDT) ולהשתמש בה על מנת לכתוב מקרי בדיקה.

הציעו 4 מילות מפתח (keywords) שמן הסתם יהיו בתשתית האוטומציה. תנו הסבר קצר מה כל מילת מפתח מבצעת.

ג) כתבו תסריט הממש את מקרה הבדיקה שכתבתם בסעיף (א) תוך שימוש בכל ארבעת מילות המפתח. **שימו לב**: זה רומז שמקרה הבדיקה שכתבתם בסעיף א' צריך להיות כזה שדורש לפחות ארבע מילות מפתח...

הערות:

- אין צורך להגדיר מילות מפתח עבור שלבי ה-setup. מקרה הבדיקה שאותו תכתבו עם KDT מתחיל משלב ה"**Steps**"
- מותר להציע גם יותר מארבע מילות מפתח אם זה עוזר.

10) [10]
נתונה הפונקציה הבאה:

```
def foo(myList):

       a = myList[3]
       b = myList[4]*myList[2]

       ret = bar(a, b)

       if ret == 0:
              myList[1] = 2*a


       if ret == 1:
           myList[1] = 2*a
```

הערה: בתשובות ה syntax המדויק ממש לא חשוב.

א) כתבו בדיקת יחידה (unit test) שתגלה את הבאג הטריוויאלי שיש ב-()foo
ב) כתבו שני stubs עבור ()bar , שידרשו לצורך בדיקות יחידה
ג) כתבו mock עבור ()bar , שידרש לצורך בדיקות יחידה

פתרונות

1) טבלת מעברים (אופציה א')

S1: Empty spot
S2: Waiting for ACK(Full) from base station ("ACK" = Acknowledge = אישור)
S3: Full spot
S4: Waiting for ACK(Empty) from base station

	S1	S2	S3	S4
S1	Any ACK from base station / NULL Message from another station / Forward message	Car entry / SendMessage(Full)		
S2		ACK(Empty) from base station / NULL Timeout / SendMessage(Full) Message from another station / Forward message	ACK(Full) from base station / NULL	Car exit / SendMessage(Empty)
S3			Any ACK from base station / NULL Message from another station / Forward message	Car exit / SendMessage(Empty)
S4	ACK(Empty) from base station / NULL	Car entry / SendMessage(Full)		ACK(Full) from base station / NULL Timeout / SendMessage(Empty) Message from another station / Forward message

טבלת מעברים (אופציה ב')

S1: Static state
S2: Waiting for Ack from base station

	S1	S2
S1	Any ACK from base station / NULL Message from another station / Forward message	Parking spot status change (from full to empty, from empty to full) / Send Message(spot state)
S2	ACK(correct spot state) from base station	ACK(incorrect spot state) from base station / NULL Timeout / Send Message(spot state) Message from another station / Forward message Spot state change / SendMessage(spot state)

2) מכונית ראשונה (נכנסת; יוצאת) כשכל החניות האחרות ריקות
מכונית אחרונה (נכנסת; יוצאת) כשכל החניות האחרות תפוסות

חניה במשבצת הקרובה ביותר; הרחוקה ביותר מתחנת הבסיס
תפיסת כל החניות בבת אחת
שחרור כל החניות בבת אחת
קבלת האישור מתחנת הבסיס לאחר 15 שניות פלוס \ מינוס אפסילון (על פי הדיוק של המערכת)

3) רכב נכנס לחניה ריקה הרחוקה מרחק של 8 חניות מתחנת הבסיס. לאחר ארבע שניות הרכב יוצא מהחניה.
=> התוצאה היא שלאחר שמונה שניות יתקבל אישור על "תפוס", בזמן שהחניה כבר פנויה. צריך לבדוק שבמקרה זה תחנת הקצה לא שולחת תשדורת נוספת כשמגיע האשור על "תפוס" אלא מתעלמת ממנו ומחכה לאישור המעודכן.

4) יכולות עזר לבדיקה בנקודת הקצה:
א) אפשרות להקטנת זמן ה – timeout לערך נמוך משניה, כך שכל פעולת חניה תייצר timeouts.
ב) גרימה לכך שגם כשהתקבל אישור נכון, הקוד שמקבל את התשדורת לא יעצור את הטיימר שמודד 15 שניות. כתוצאה מכך ה- timeout יופעל.

יכולות עזר לבדיקה בתחנת הבסיס:
א) עיכוב של משלוח האישור. התוצאה תהיה שנקודת הקצה תחווה timeout.
ב) שליחה של אישור לא נכון (כלומר, כשמגיעה הודעה על "חניה תפוסה", ישלח אישור על "חניה ריקה" וההפך. זה יגרום לנקודת הקצה להתעלם מהאישור ובסופו של דבר להגיע ל timeout
ג) שליחה של מספר לא נכון באישור. למשל, עבור הודעה שהתקבלה מתחנה n , לשלוח אישור לתחנה n+1. מבחינת נקודת הקצה, האישור לא נשלח.

5) דוח באג: (גם דוחות בעברית זה בסדר...)

Title: The end point modifies the transmission from the neighboring end point before repeating it

Description:
Per the requirements, when an end point receives a message from a neighboring end point, it should repeat it unmodified. In the current SW release, the edge point turns the status of the parking spot in the message: if the message indicates a full spot, it is repeated with an empty spot indication; if the message indicates an empty spot, it is repeated with a full spot indication.

Step to reproduce
1) Select a parking spot that fits the following:
   a. The spot is NOT the parking spot right next to the base station.
   b. The number of parking spots between the selected spot and the base station is odd (not counting the selected parking spot)
2) Park a car in the selected spot
3) Review the message as it arrives to the base station

Expected result
The base station receives a message from the selected parking spot, indicating the spot is full

Actual result

The base station receives a message from the selected parking spot, indicating the spot is empty

Additional information
The bug can only be reproduced if the selected parking spot has an odd number of parking spots between it and the base-station. Otherwise, as each end-point flips the message, the correct message arrives to the base station.

Suggested Severity: Critical

Suggested Severity Reason: This bug means the status of half the parking spots in the base-station's database is incorrect

6) הצעות שיתקבלו:
- הרבה מכוניות נכנסות ויוצאות מהחניה תוך פרק זמן קצר
- שינוי מצב חניה אחד במהירות גבוהה (פחות משניה ביו השינויים)

יתכן שיהיו הצעות נוספות שלא חשבני עליהם וכן נקבל אותם.

להלן דוגמה למקרה בדיקה אפשרי. יתקבלו גם מקרי בדיקה שמבוססים על סימולטורים עבור הרגשים, על מנת לסמלט שינויים בקצבים גבוהים מאוד. למעשה, כל הצעה הגיונית וכתובה היטב תתקבל.

כתבתי כאן מקרה בדיקה שלם (כי בהתחלה זו היתה השאלה) אז אני משאיר אותו. מבחינת המבחן – מספיק מטרה ושיטת הבדיקה.

כותרת: שינוי מצב כל החניות בקצבים גבוהים

מטרה: מדידת יכולת המערכת לעמוד בשינויים מהירים במצב החניה של תחנות קצה רבות

שיטת הבדיקה: מול כל נקודת חניה מציבים רכב. כל רכב נכנס ויוצא מהחניה בקצב המקסימלי האפשרי (בנסיעה קדימה ואחורה). הבדיקה נמשכת שלוש דקות.

SETUP:

העמד מול כל נקודת חניה מכונית ובה נהג. השתמש במכוניות שונות (קטנות; גדולות; כבדות וקלות)

TEST:

צעד	תוצאה מצופה
כל נהג נכנס ויוצא מהחניה שמולו בקצב המירבי האפשרי	
המשך בפעולה למשך 3 דקות	
לאחר שלוש דקות, כל נהג עוצר בנקודה בה הוא נמצא (בתוך החניה, מחוץ לה או באמצע)	בסיס הנתונים של החניות (בתחנת הבסיס) מראה נכון את מצב החניות
	המערכת לא נתקעה או קרסה
חזור על הבדיקה שלוש פעמים	

7) הקוד לא מוודא שאורך str שהועבר לפונקציה הוא 4 בתים. כלומר, אפשר לכתוב כל str שהוא וכאשר עוברים את הבדיקה על 'Free' או 'Used', הוא יועתק אל בסיס הנתונים. בדרך הוא גם יגרום ל - buffer overflow בשורה המעתיקה את str ל - buffer .

8)

:Dumb Fuzzing

SetStatus(123, "wefdsfdsfdf")
SetStatus("x", "1-%fd3hgF")

כללית: מנגנון ה-fuzzing ייצור קלטים רנדומיים עבור str ו – slot. הערכים שיבחרו ל-slot לא מתחשבים בבדיקה הראשונית שמסננת קלטים שבהם slot לא בתחום המותר. הערכים שיבחרו ל-str יהיו ללא הגבלה.

:Smart Fuzzing

SetStatus(MAX_SLOT_NUMBER, "sfdf")
SetStatus(7, "FreeAAA")

כללית: מנגנון ה-fuzzing ייצור קלטים סמי-רנדומיים עבור str ו – slot. הערכים שיבחרו ל- slot מתחשבים בבדיקה הראשונית שמסננת קלטים שבהם ה- slot לא בתחום המותר. כלומר, יבחרו ערכים בין 1 ל-MAX_SLOT_NUMBER. הערכים שיבחרו ל-str יהיו ללא הגבלה.

9)

A)

Note: The scenario described here is more complicated and long than what would be considered a perfect answer**.**

Setup and Pre-conditions

1. The user U1 has at least 2 contacts in WhatsApp: C1 and C2
2. Assumption: the phone is connected to the network without any data limitations
3. Assumption: the automation system has a connection to all 3 users' phone and can interact with their WhatsApp application via some protocol to do actions, get status, see what was received etc.

Steps	**Expected Results**
Send "Hello" from U1 to C1	U1's message arrived to C1
C1 reads message	Read indicator on U1's WhatsApp shows C1 read the message
C1 sends "Hi" to U1	C1's reply arrived

U1 forwards C1's message to C2	U1's message arrived to C2
C2 does not read the message	U1 read indicator shows C2 did not read the message yet
C2 deletes the message	Message deleted on C2's WhatsApp
	U1 read indicator shows C2 did not read the message yet

B)

// Send the text in "Message" from user "From" to user "To"
SendMessage From, To, Message

// Read the last message arrived from contact "From" to user "To" and verify its text to be "Message"
ReadMessage From, To , Message

// Check read indicator on the last message sent from user "From" to contact "To" and verify it is equal to "ReadStatus" (read / not read)
CheckReadIndicator From, To, ReadStatus

// Forward the last message arrived to user "User", from user "From" to contact "To"
ForwardMessage User, From, To

// Delete the last message received by user "User" from user "From"
DeleteMessage User, From

C)

Assumption: The automation system adds 3 seconds delay between each command, to allow for network traffic delays. The tests are ran in an environment where 3 seconds are more than enough to complete the transaction.

SendMessage U1, C1, Hello

ReadMessage U1, C1, Hello

CheckReadIndictor U1, C1, Read

SendMessage C1, U1, Hi

ReadMessage C1, U1, Hi

ForwardMessage C1, U1, C2

CheckReadIndictor U1, C2, NotRead

DeleteMessage U2, C2

CheckReadIndictor U1, C2, NotRead

(10

א) הבאג הטריוויאלי הוא שאין בדיקה שמוודאת שבמערך myList יש בכלל חמש אברים, לפני השימוש בהם.

כיוון ש-foo() לא מחזיר ערך, כל שיבדק כאן זה שהטסט לא התמוטט (מה שבפועל יקרה כי ההרצה תגרום לשגיאה של array index out of range).

```
def test_foo_rejects_array_shorter_than_five()
     testList = []
     foo(testList)
     assert True
```

ב)

```
def bar_stub_ret0():
     Return 0

def bar_stub_ret1():
     Return 1
```

ג)

Assumption: foo() was called with myList = [1,7,5,23,9]
FAIL is defined as False
PASS is defined as True

```
def bar_mock(a, b):
     gTest_result = FAIL
     if (a == 23 && b == 45):
          gTest_result = PASS
     return 0  # Could just as well be return 1 .
```

Additional explanation (not requested in the exam): A test can now:

- Set the test framework to call bar_mock() instead of bar()
- Call foo() with myList = [1,7,5,23,9]
- Assert on gTest_result - a global variable. If it is True, than we know that the processing done in foo() in setting the values of a and b, was done correctly.