**Assignment 3 – integration, control and data flow testing**

.

**הוראות הגשה:**

**הגשה בקבוצות של 4 סטודנטים.**

יש להעלות לאתר ה-moodle קובץ zip אחד.

את הקוד של הבדיקות אנטגרציה יש להגיש בקבצי java. את הבדיקות של כל פונקציה יש להגיש בקובץ נפרד המכיל את כל ה-unit tests ו-stubs או drivers בהתאם לשיטה. למשל, עבור פונקציה merge צריך להגיש שני קבצי java: MergeTobDown.java ו-MergeBottomUp.java.

את החלק התאורטי יש להגיש בקובץ PDF אחד.

**בהצלחה!**

**חלק 1 (50 נק') – Integration testing**

נתון קובץ Program.java המכיל 14 פונקציות התלויות אחת בשנייה. מעל כל פונקציה רשום מה הפונקציה צריכה לבצע ובחלק מהפונקציות יכולות להיות שגיאות לוגיות.

עליכם לבצע בדיקות אינטגרציה של כל הפונקציות בשיטות הבאות:

1. Top down approach
2. Bottom up approach

ענו על הסעיפים הבאים:

* ציירו את ההיררכיה של הפונקציות.
* עבור כל אחת מהשיטות יש לפרט באיזה סדר בצעתם בדיקות של הפונקציות.
* עבור כל אחד מהפונקציות שבדקתם (בכל שיטה) יש לפרט בקובץ pdf את כל מקרי בדיקה ובדיקות יחידה (unit tests) שביצעתם, ואת כל ה-drivers ו/או stubs שיצרתם.
* עבור כל מקרה בדיקה (test case) יש לפרט בטבלה את הפרטים הבאים:
* שם הפונקציה הנבדקת.
* קלט של הפונקציה (אם יש).
* מה הפונקציה אמורה לבדוק?
* פלט הצפוי.
* פלט אמתי של הפונקציה.
* האם הטסט עבר או נכשל.
* יש לצרף את הקוד של הבדיקות כפי שמתואר בהוראות ההגשה.

**חלק 2 (50 נק') – Control and dataflow testing, symbolic execution**

נתונות פונקציות הבאות:

A)

**public** **static** **double** calculatePrice(**double** distance, **int** age, **int** type) {

**double** cost = 1000;

**boolean** valid = **true;**

**if** (age < 1 || age >120 || distance < 0)

valid = **false;**

**if** (valid && age < 18 && distance > 500)

valid = **false;**

**if** (valid)

**switch** (type) {

**case** 2:

**double** temp = age \* distance / 100;

cost = Math.min(temp, cost);

**if** (age < 18)

cost = 100;

**break**;

**case** 1:

cost = distance \* 7 / age;

**break**;

**default**:

valid = **false;**

}

**if** (!valid)

**return** -1;

**return** cost;

}

B)

**public** **static** **int** findMostCommonPositiveNumber(**int**[] array){

**int** commNum = -1;

**int** commNumCount=0;

**for**(**int** i=0; i< array.length; i++){

**if** (array[i]>=0){

**int** count = 1;

**for**(**int** j=i+1; j< array.length; j++){

**if** (array[i] == array[j])

count++;

}

**if** (count>commNumCount){

commNumCount = count;

commNum = array[i];

}

}

**else**

System.***out***.println("Negative number!");

}

**if** (commNum!=-1)

System.***out***.println("The common number is: "+commNum);

**else**

System.***out***.println("No positive number in the array");

**return** commNum;

}

עבור הפונקציות A ו-B ענו על הסעיפים הבאים:

1. בנו את הגרף CFG עבור הפונקציות.
2. האם קיימים מסלולים שלמים (complete path) לא פיזיביליים בגרף? אם כן, רשמו לפחות מסלול אחד.
3. מצאו מסלולים שלמים העונים על הקריטריון של statement coverage מלא. רשמו את הקלט המתאים (test case) עבור כל אחד מהמסלולים. אם לא קיימים מסלולים פיזיביליים הנותנים כיסוי מלא, הסבירו מדוע.
4. מצאו מסלולים שלמים העונים על הקריטריון של branch coverage מלא. רשמו את הקלט המתאים (test case) עבור כל אחד מהמסלולים. אם לא קיימים מסלולים פיזיביליים הנותנים כיסוי מלא, הסבירו מדוע.

עבור הפונקציה A ענו על הסעיפים הבאים:

1. בנו את הגרף DFG עבור הפונקציה.
2. עבור כל definition של כל אחד מהמשתנים ציינו האם הוא global definition. הסבירו.
3. רשמו את כל ה-du pairs של כל אחד מהמשתנים. כלומר, עבור כל משתנה, יש לרשום את כל הזוגות של definition ו- use (p-use או c-use) כך שיש def clear path בין definition ו-use.
4. מצאו קבוצת מסלולים וקלט מתאים לכל מסלול, אשר תהוונה כיסוי מלא לפי קריטריונים הבאים:
   1. All-defs
   2. All-C-uses
   3. All-P-uses
   4. All-uses
   5. All-P-uses/Some-C-uses
   6. All-C-uses/Some-P-uses

אם לא ניתן לעמוד באחד או יותר מהכיסויים הסבירו למה.

1. נתון הקוד הבא המתבסס על קוד A עם שינויים:

**public** **static** **double** calculatePriceAdjusted(**double** distance, **int** age) {

**2. double** cost = 1000;

**3. if** (age < 1 || age >120 || distance < 0)

4. cost = age\*(Math.E^)(-1)\*distance();

**5. else if** ( distance > 500)

6. cost = Math.min(age \* distance / 100, cost);

**7.** assert (cost<=1)

}

ציירו תרשים עץ symbolic execution סטאטי המפרט את ה- σו- π עבור כל stmt.