**מכוןן לבדיקה של הבגרות הפנימית – נכתב ע"י ערן סלומון**

פרטים טכניים יבשים:

* הבגרות הפנימית מיועדת לאלו שבחרו בכך, במסגרת ההנחיות שניתנו עקב מגפת הקורונה.
* במקצוע מדעי המחשב, במקום בחינה, ניתנה מטלה להגשה, אשר בה יש שני חלקים
  + חלק א': בדיקות אוטומטיות – יבוצעו ע"י קוד שנכתב במיוחד לכך ויפורסם לתלמידים.
    - תלמידים הניגשים לבגרות חיצונית ויחליטו להגיש מטלה זו לשם התרגול – נריץ את הבדיקות שלנו, ללא חלק של הבדיקה הפרונטלית כפי שיפורט בהמשך.
  + חלק ב': בדיקה פרונטלית, בחלק זה התלמיד/ה י/תשב עם המורה (לא בהכרח המורה המשויך) אחד על אחד עם הקוד שנכתב ע"י אותו תלמיד/ה, בחלק זה המורה ישאל שאלות על הקוד, ועל התלמיד לענות על השאלות הרלוונטיות. השאלות שישאלו יכולות להיות בעלות קשר לקטעים שנכשלו, או קטעים שהצליחו, וכן הן ישתנו מתלמיד לתלמיד. כל הבדיקות הפרונטליות יערכו ביום 14.5 – אותו היום בו נערכות כל הבגרויות הפנימיות. מצופה מהתלמיד/ה לענות באופן מלא על כל השאלות.
* מסמך זה מפרט על אופן הבדיקות האוטומטיות, וכן מהווה מאגר שאלות לחלק הבדיקה הפרונטלית.

בכל מקום בהמשך המסמך בו רשום "תלמיד" – הכוונה גם לתלמידה, נכתב מטעמי נוחות בלבד.

חלק א': בדיקות אוטומטיות

**אופן הבדיקה:**

עבור כל השאלות נכתבו בדיקות "טסטים" שמטרתם לבחון את נכונות הקוד. הטסטים ישלחו קלטים ספציפיים מתוך מטרה לקבל פלט מסוים. במידה והפלט שמתקבל זהה לפלט המצופה – תצא בודעה (בצבע ירוק) שהתלמיד עבר את הטסט, והקוד יזכה אותו בניקוד בהתאם למכוון המפורט פה. במידה והפלט שמתקבל שונה מהפלט המצופה, ו/או התקבלה שגיאה Exception, הטסטים ייכשלו, תצא הודעה בצבע אדום המסבירה על מה שאירע והתלמיד לא יקבל ניקוד כלל. יש לציין שאם תוכנית רצה ללא שגיאה, תודפס הודעה מתאימה בצבע ירוק, ואם התוכנית קיבלה שגיאה, תודפס הודעה בצבע אדום. בסוף כל משימה, יודפס בצבע כחול הניקוד שקיבל אותו תלמיד על חלק זה. בכוונה החלטתי שבסוף כל בדיקת משימה התוכנית תעצור ותחכה ללחיצה enter מהמשתמש, זאת כדי שנוכל בתור המורים שמעריכים לראות האם היו שגיאות, ונוכל לתכנן עבור אותו תלמיד שאלות לחלק הפרונטלי.

**שאלה 1:**

בקוד נדרש גם להחזיר וגם להדפיס. לצערי לא מצאתי דרך טובה להשוות באופן אוטומטי את ההדפסה, ולכן נצטרך לבצע השוואה ידנית.

שלחתי שני מערכים בגודל 62, שניהם סדרה חשבונית עם . ההבדל הוא באיבר הראשון (וכל האיברים שאחריו כתוצאה מכך). במערך הראשון בעוד במערך השני האיבר הראשון הוא . ההבדל הוא במערך השני, בו אחד האיברים הוא 669, ולא אמור להיכלל בספירה (אם נכלל וקיבל 30 במקום 29, כנראה עשה גדול שווה במקום גדול).

עבור ההדפסה, כפי שציינתי, אין דרך לבדוק בצורה אוטומטית, ולכן צבעתי בצבע Cyan את ההדפסות שיצאו במהלך ריצת התוכנית. לאחר מכן התוכנית תשאל האם הפלט יצא זהה לזה שציפינו לו. יש לענות בעזרת y או n. כל קלט אחר ידרוש להזין מחדש.

עבור ערך מוחזר נכון – ניתנה נקודה אחת. עבור ערך מודפס נכון – ניתן נקודה אחת. כפול 2 מערכים – סה"כ 4 נקודות.

**שאלה 2**

בשאלה זו שלחתי 10 מערכים, כאשר המערך הריק צריך להחזיר true, מערך בגודל אחד גם צריך להחזיר true, ועוד מספר מערכים נוספים. לכל מערך שנשלח התוכנית תזכה בחצי נקודה במידה והוא החזיר את התוצאה הרצויה. סה"כ – 5 נקודות.

**שאלה 3**

בשאלה זו בניתי מחלקה בשם MySofa – זוהי מחלקה שמהווה את הפיתרון שלי למחלקה זו – ואני משתמש בבנאים שלה על מנת לבנות עצמים כאלו בעצמי. בניתי בנוסף פעולה שמשווה בין עצם Sofa לעצם Mysofa.

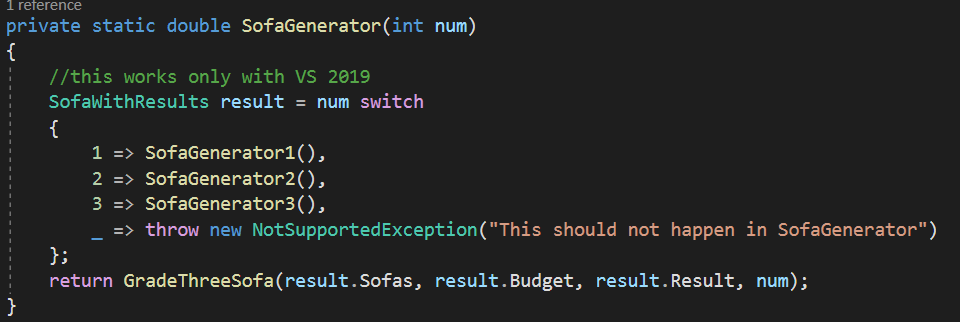
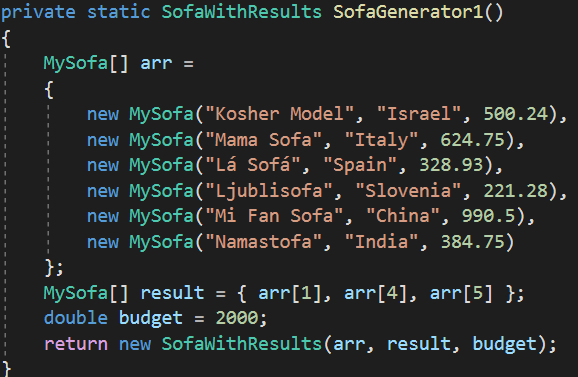
לסעיף א' – שנועד לבדוק תקינות של בניית מחלקה בסיסית. בניתי שתי עצמים:

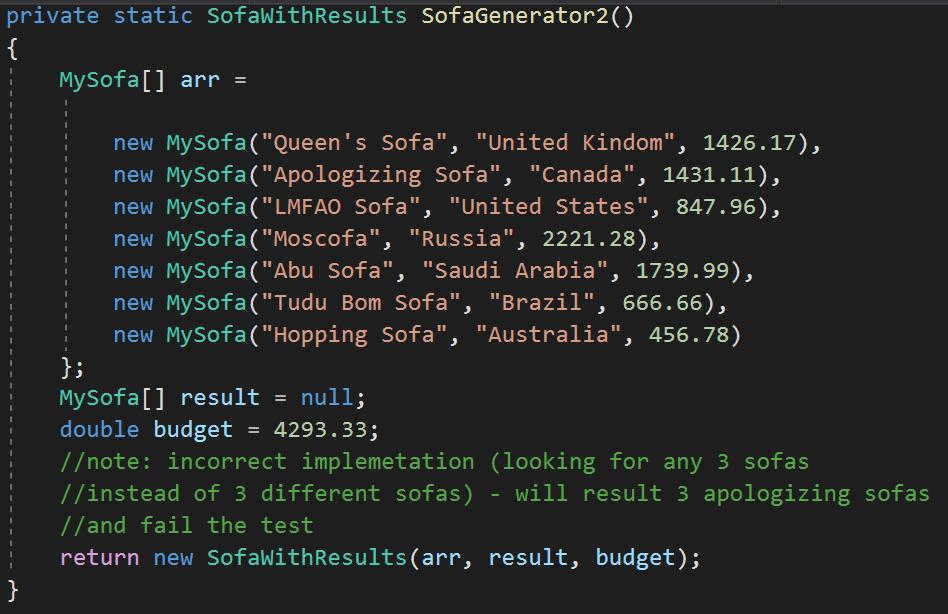
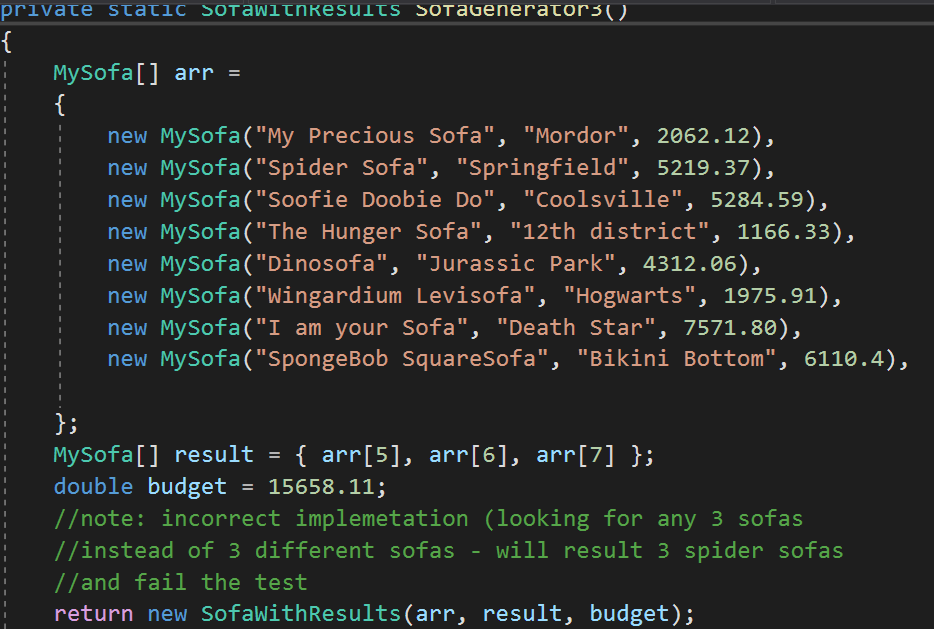
 

התוכנית תבנה עצם מסוג Sofa ותבדוק שזה מצליח בלי בעיות, וכן שפעולות Get עובדות גם כן. לכל בנייה מוצלחת – 0.75 נקודה, סה"כ 1.5 נקודות לסעיף א'.

עבור סעיף ב:

לצורך זה בניתי מבנה נתונים (שרעיונות דומים שלו יחזרו על עצמם במשימות נוספות) בשם SofaWithResult, המכיל בתוכו מערך MySofa של מה שאני שולח, משתנה double budget, ומערך נוסף של MySofa לתוצאה הרצויה. המטרה לכך הינה שאוכל לאגד בקלות את כל הקלט והפלט לכדי מבנה נתונים אחד, ובכך אוכל ליצור תבנית שתייצר עבורי קלטים ופלטים צפויים ותחזיר את הכל יחדיו.

לאחר מכן – התבנית שלי כוללת את הדבר הבא: אני קורא לפעולה בשם SofaGenerator עם מספר כלשהו בין 1 ל – 3. כל אחד כזה קורא לפונקציה נוספת שיצרתי שיוצרת מערך ספות, budget ופלט. כך זה נראה:



אציין שבדקתי את ענייני שגיאות עיגול, והסכום נבחר כך שלא אמורה להיווצר שגיאה כזו. למרות זאת הבדיקה בודקת אם ההפרש בערך מוחלט קטן מ-0.01.

כל בדיקה שעוברת בהצלחה תזכה ב – 1.5 נקודות, סה"כ 4.5 נקודות לסעיף ב'. ו-6 נקודות לכל השאלה.

כמו כן, הטסטים מעט מכשילים, כפי שרשום בהערות של 2 ו 3. מימוש בצורה שבה מחפשים ספות עם כפילויות (כלומר מהסוג i=0 j=0 k=0, בניגוד להוראות השאלה), אז הטסטים 2, 3 ייכשלו, אבל הראשון יעבור. לכן כדאי להסתכל על התוצאות: אם בטסט 2 נקבל 3 ספות מסוג Apologizing Sofa ובטסט 3 נקבל 3 טסטים מסוג Spider Sofa, סימן שהלולאות שהתלמיד עשה התחילו מ-0 כל פעם. אפשר להשתמש בזה כדי שבבדיקה הפרונטלית נוכל לשאול אותם למה זה קרה ולראות אם הם עולים על זה.

**שאלה 4:**

המטרה בשאלה היא ליצור מימוש שעונה **במדויק** על ההדפסות המתוארות לעיל, ולכן הטסטים שלי מתמקדים בדיוק בהדפסות הללו. בשאלה יש 8 הדפסות, ולכן כל הדפסה נכונה תזכה ב 0.75 נקודות, סה"כ 6 נקודות.

במידה ואחד מתוך שלושת המשקולות שבנינו זורק שגיאה בפעולה בונה, כל התרגיל מוכשל, אין בדיקה של ההדפסות כלל, והניקוד לשאלה זו יהיה 0.

למרות שהכנסתי הערה בקבצים שניתנו לתלמדים שעליהם להדפיס במדויק, בבדיקה שלי, על מנת למנוע הכשלות ב ToString() עקב בלבול בין אותיות גדולות או קטנות ו/או רווחים – הפעלתי על התוצאה את הפונקציה ToLower() וכן פונקציה שמצאתי ב StackOverflow שמורידה את כל הרווחים, ואת התוצאה השוויותי לזו שאמורים לקבל ע"פ השאלה, לאחר הפעלת שתי הפונקציות הללו גם כן.

**שאלה 5:**

בשאלה זו יצרתי מבנה נתונים של רשימות הדומה ברעיון שלו לזה של שאלה 3. מבנה הנתונים כלל את הרשימה המקורית, הרשימה במקומות הזוגיים, הרשימה במקומות האי זוגיים והרשימה הנוצרת כתוצאה מהפעלת SwitchChain.

גם כאן, במידה ומקבלים על אחת מהרשימות שגיאה בפעולה בונה, לא יורצו כלל הבדיקות המיועדות עבור אותה רשימה והניקוד יהיה 0 מתוך 4.

לכל רשימה 4 פעולות שצריך לבדוק, כל פעולה שווה נקודה, סה"כ 8 נקודות.

יצרתי 2 רשימות:

0-->1-->2-->3-->4-->

זוגי: 0-->2-->4-->

אי זוגי: 1-->3-->

מוחלף: 1-->0-->3-->2-->4-->

ואת הרשימה:

0-->1-->2-->3-->

זוגי: 0-->2-->

אי זוגי: 1-->3-->

מוחלף: 1-->0-->3-->2-->

**שאלה 6:**

בשאלה זו יצרתי מבנה נתונים הדומה לזה שיצרתי בשאלה 3 עבור האחסון של המחסנית והתוצאה.

עבור סעיף א' – בדקתי עם 2 אפשרויות: המחרוזת "0" והמחרוזת "567" – כל הצלחה מביאה 0.5 נקודה, סה"כ 1 נקודה. **חשוב – בבדיקה הפרונטלית יש לבדוק שהמימוש של זה רקורסיבי ע"פ ההוראות, וכן שאין שימוש ב int.Parse(). תלמיד שעושה כך, יש להוריד לו 5 נקודות מהפרונטלי, אלא אם התלמיד קיבל ציון נמוך בבדיקות האוטומטיות והראה רצינות בפרונטלי, ואז ניתן להוריד פחות, לשיקול דעת המורה (אבל לא לוותר לו בכלל כי בכל זאת סטה מההוראות).**

עבור סעיף ב' – בדקתי את ארבעת תרגילי החשבון הבאים: 247+669=916, 149-567=-418, 2673\*1597=4268781, 15684/221=70. כל הצלחה מביאה 0.5 נקודה ובסה"כ 2 נקודות.

עבור סעיף ג' – בדקתי 5 מחסניות (ראש המחסנית מצד ימין):

[9,1,2,3,+,4,2,5,6,7,=,3,3,3] – המחזירה 51690

[5,7,1,3,-,3,5,7,0,=] – המחזירה 2143

[5,3,2,\*,1,2,1] – המחזירה 64372

[3,4,4,5,/,5,6,9,=,1] – המחזירה 6

[9,8,7,6,5,4,3,2,1,\*,0] – המחזירה 0

כל הצלחה מביאה נקודה ובסך הכל 5 נקודות.

**יש לבדוק בסעיף ג' שאין קריאה לסעיף א', ובמידה ויש צריך להוריד ניקוד מהפרונטלי.**

סה"כ לשאלה זו 8 נקודות.

**שאלה 7:**

גם פה יצרתי מבנה נתונים שמאגד בתוכו את העץ ואת התשובה הצפויה, יצרתי 8 כאלה, 4 לסעיף א' ו 4 לסעיף ב'. כל בדיקה = נקודה וסה"כ 8 נקודות. ההדפסות המוצגות כאן הן בשיטת InOrder

סעיף א':

1. [8] – true
2. [6 8 5] – false
3. [6 8 5 2 4] – true
4. [2 6 4 4 5 21 5 3 9] – false

סעיף ב':

1. [8] – true
2. [6 9 3 2 3 1] – true
3. [6 8 5 2 4] – false
4. [2 6 4 4 5 20 5 3 9] – false

טסטים 7 ו-8 הם על עצים שהם סטטיים אבל לא סטטיים מאוד.

**שאלה 8:**

קודם כל – בבדיקה הפרונטלית חייבים לבדוק שזה מקיים את כללי ההורשה שלמדנו בכיתה. תלמידים שלא מימשו את ההורשה בצורה מיטבית יש להוריד 2 עד 10 נקודות בהתאם לחומרת המימוש הלא נכון ובהתאם לשיקול הדעת של המורה בנוגע לתוצאות הבדיקה האוטומטית ורצינות התלמיד.

הבדיקות האוטומטיות מתמקדות בעיקר בפונקציונלית של המחלקות הללו, על איכות הקוד, הבחירה בהורשה, וכו' יש לבחון כחלק מהבדיקה הפרונטלית

לכל מחלקה מבין 3 המחלקות Car, Bus, Bicycle – יצרתי עצם ובדקתי שהמימוש שלו הגיוני. כל אחד 5 נקודות.

במידה ויצירת העצם נכשלה בפעולה הבונה, לא תיבדק הפונקציונליות שלה, והתלמיד יקבל על מחלקה זו 0 מתוך 5.

כמו כן, את ה ToString() יש לבדוק באופן ידני. תוצג הודעה המתארת את מצבו של העצם, ותוצג ההחזרה של התלמיד מ ToString בצבע Cyan, וכמו בשאלה 1, יש להשיב באמצעות y או n האם זה תואם.

עבור car החלוקה הייתה באופן הבא:

כל Get – 0.25 נק' כפול 6 = 1.5 נק'

ToString() – 0.3 נק' כפול 1 = 0.3 נק'

מכירת רכב – 0.2 נק' כפול 1 = 0.2 נק'

SetAmount: יש 3 בדיקות, השניים הראשונות הן מחוץ לטווח ה capacity ולכן צריכות להיות ללא שינוי והשלישית אמורה לשנות את ה amount.

ללא שינוי – 0.4 נק' כפול 2 = 0.8 נק'

עם שינוי – 0.2 נק' כפול 1 = 0.2 נק'

FuelUp() – 1.5 נק' כפול 1 = 1.5 נק'

בדיקת amount לאחר מכן: 0.5 כפול 1 = 0.5 נק'

עבור bus החלוקה הייתה באופן הבא:

כל Get – 0.15 נק' כפול 7 = 1.05 נק'

ToString() – 0.2 נק' כפול 1 = 0.2 נק'

מכירת רכב – 0.25 נק' כפול 1 = 0.25 נק'

שינוי קו אוטובוס: 0.5 כפול 1 = 0.5 נק'

SetAmount: יש 3 בדיקות, השניים הראשונות הן מחוץ לטווח ה capacity ולכן צריכות להיות ללא שינוי והשלישית אמורה לשנות את ה amount.

ללא שינוי – 0.4 נק' כפול 2 = 0.8 נק'

עם שינוי – 0.2 נק' כפול 1 = 0.2 נק'

FuelUp() – 1.5 נק' כפול 1 = 1.5 נק'

בדיקת amount לאחר מכן: 0.5 כפול 1 = 0.5 נק'

עבור bicycle החלוקה הייתה באופן הבא:

כל Get – 0.3 נק' כפול 5 = 1.5 נק'

ToString() – 0.5 נק' כפול 1 = 0.5 נק'

מכירת רכב – 0.5 נק' כפול 1 = 0.5 נק'

Ride כך שלא נגמרת הסוללה - 0.25 נק' כפול 1 = 0.25 נק'

Charge לא מעבר ל 100% - 0.25 נק' כפול 1 = 0.25 נק'

Ride כך שנגמרת הסוללה באמצע - 1 נק' כפול 1 = 1 נק'

Charge מעבר ל-100% - 1 נק' כפול 1 = 1 נק' (הטסט מעוצב ככה שהוא ייכשל ככל הנראה אם לא עבר את הטסט שלפני עקב כך שאחוז הסוללה התעדכן אל מתחת ל-0)