תיעוד חיצוני-את"מ 4

תאריך: 30/6/2017

**סהר כהן: 206824088**

**יובל נהון: 206866832**

**הסבר על התוכנית:**

התוכנית משמשת ממשק אינטראקטיבי, דמוי משחק, להשלמה אוטומטית של מילים והרכבת הודעה בעזרתן. לתוכנית תתווספנה בזיכרון רשימה מקושרת **Dict** המכילה מחרוזות ממוינות לקסיקוגרפית, ולה מספר תאים ריקים (תאים בהן מחרוזת ריקה 0) לצורך השלמת המשפט, תווית **X** המייצגת את מספר הטעויות המותר להשלמות המילים בריצת התוכנית ותווית **rate** אשר מגדירה את מספר תקתוקי השעון בשנייה (מספר דצימאלי). השלמה של מילה לפי X טעויות מתבצעת משיקולי החסרת אות, הוספת אות או החלפת אות ממילת הקלט (התחילית) X פעמים כך שתוכל להוות תחילית של מילה במילון. בתחילת הריצה, המשתמש בוחר את הזמן המוקצב לו לכתיבת ההודעה ואת גודלה. לאחר מכן, המשתמש חופשי לכתוב מילים כך שבסוף כל מילה יקיש אחד מן התווים הבאים: '!', ' ' (רווח). אם המשתמש הקיש רווח – המילה שכתב תתווסף להודעה ולמיליון. אם המשתמש הקיש '!', אז התוכנית תחפש במילון מילים אשר המילה שכתב מהווה תחילית שלהן, עד X טעויות. סדר הצגת המילים הוא לפי מספר הטעויות (מתחילית מושלמת ועד תחילית עד כדי X טעויות), בסדר לקסיקוגרפי. המשתמש רשאי לבחור את אחת מן האפשרויות המוצעות לו להשלמה ואז המילה תתווסף להודעה; או לא לבחור באף אפשרות (או להקיש '!' שוב ולהמשיך לכתוב את המילה הנוכחית).

התוכנית תעצור כאשר נגמר הזמן **או** כשנקלט מספר המילים הדרוש **או** כאשר המשתמש הקליד '.' במהלך הרכבת אחת המילים להודעה. בנוסף, בסוף ריצת התוכנית ימצאו במילון כל המילים אשר נמצאו שם בתחילת ההרצה, בתוספת למילים החדשות שהמשתמש הוסיף, **בסדר ממוין**.

**הבהרה:** מיון המילים הוא לפי **כתובת** המחרוזות במימוש זה, ולא לפי הערך הנמצא במחרוזות!

**אופן פעולת התוכנית:**

**main** – ראשית, התוכנית מאתחלת את המחסנית, מחשבת את גודלו ההתחלתי של המילון הנתון על ידי השגרה **dictCalcLen**, וקוראת לשגרת עזר נוספת **startGame** על מנת להציג את ההדפסות ההתחלתיות למשתמש ולקלוט את זמן הריצה ואת גודל ההודעה הרצויים בריצה זו. בשלב זה אנו נחכה לקלט מהמשתמש, ולכן נפעיל את השגרה **scanEnable** לצורך הדפסת התווים "> " המדגישים למשתמש שרשאי להקליד. כעת, התוכנית מגדירה בווקטורי הפסיקה את שגרת שירותי הפסיקה של השעון והמקלדת, **clock** ו – **msgScan** בהתאמה, כאשר עדיפות הפסיקה של השעון היא 6, ועדיפות הפסיקה של המקלדת היא 4. כעת התוכנית תחכה לפסיקה מהשעון או המקלדת, ובזמן ההמתנה תבצע את הדברים הבאים: תבדוק האם נגמר הזמן, תבדוק האם נקלטו מספר המילים שהמשתמש הגדיר ותבדוק האם במהלך פסיקת המקלדת האחרונה הודפס התו '.' . אם אחד התנאים מתקיימים: התוכנית תסיים את ריצתה ותדפיס הודעה מתאימה, ואת ההודעה אשר נקלטה במהלך ריצת התוכנית. אחרת, תבדוק האם גודל המילון הוא 0. אם כן, תחזור לבצע את הבדיקות באופן מחזורי בזמן שמחכה לפסיקות. אחרת, תקרא לשגרת **RevBubble** אשר ממיינת את המילון בהנחה והוא כבר ממוין, פרט אולי למחרוזת באיבר האחרון שלו שאינו ריק. התוכנית תמשיך לבצע את בדיקות אלו ולמיין את הרשימה עד שאחת הבדיקות מתקיימת, או כאשר התקבלה פסיקה.

לצורך הגשמת משימת התוכנית – נכתבו מספר רב של שגרות:

**הערה: רגיסטר הקישור בכל השגרות שאינן שגרות שירות של פסיקה בתוכנית זו הוא pc !**

**dictCalcLen** – מקבלת את המילון ואת הכתובת בה יישמר גודלו in stack, ומשנה את הערך היושב בכתובת הרצויה בגודל המילון ההתחלתי על ידי מעבר על המילון מהאיבר הראשון ועד האיבר האחרון אשר מכיל ערך 0, או איבר אשר מצביע לכתובת 0 בתור איבר הבא אחריו.

**putChar** – מקבלת תו במחסנית ומדפיסה אותו בצורת משאל.

**printSentence** – מקבלת מחרוזת במחסנית אשר נגמרת ב terminator '@', ומדפיסה את כל התווים במחרוזת זו עד ה – '@' באמצעות **putChar**.

**endLine** – מדפיסה ירידת שורה.

**scanEnable** – מדפיסה את תווי ההרשאה לקליטת קלט: `> ` (עם רווח).

**readInput** – קוראת בצורת משאל קלט מהמשתמש עבור אחד מהשדות: זמן או אורך הודעה. מקבלת במחסנית את הכתובת בה יישמר הקלט שהמשתמש מכניס. המילה המורכבת מומרת למספר ונשמרת בכתובת אשר התקבלה במחסנית. השגרה תומכת בקבלת backspace ותרשה הדפסה של תו זה (אלא אם המשתמש מנסה למחוק את התווים `> `), ותשנה את המספר הנקלט עד כה בהתאם. סוף קליטה על ידי הקשה על enter.

**removeLastLetter** – נקרא לשגרה זו לאחר הדפסה של backspace שנקלט מהמשתמש: השגרה תדפיס רווח ואז שוב backspace על מנת למחוק את התו האחרון.

**startGame** – שגרה זו מבצעת את ההדפסות הראשונות של התוכנית (קליטת גודל הודעה, קליטת זמן ריצה והדפסת התחלת המשחק) ותקלוט לתוך השדות המתאימים בזיכרון את אורך ההודעה ואת זמן הריצה, באמצעות שימוש בשגרות **printSentence** ו – **readInput** עם הקלט המתאים, והשגרה **endLine**.

**gameResults** – מדפיסה את ההודעה הסופית והודעת סיום המשחק.

**gameOver** – מדפיסה "Thank you!" על מנת להציג למשתמש ריצה מוצלחת של התוכנית אשר נגמרה עקב קליטת כל המילים הנדרשות להודעה או הקלדה של נקודה, ואז קוראת ל **gameResults** על מנת להציג את התוצאות.

**timesUp** – מדפיסה "Sorry Time is up!" על מנת להציג למשתמש שריצת התוכנית נגמרה עקב כך שנגמר הזמן כפי שהוכנס על ידי המשתמש, ואז קוראת ל **gameResults** על מנת להציג את התוצאות.

**isLexiBigger** – מקבלת שלושה ערכים במחסנית: ערך זבל אליו תכתב תוצאת השגרה, כתובת של מחרוזת אחת וכתובת של מחרוזת שנייה. בודקת האם המחרוזת הראשונה גדולה לקסיקוגרפית מהמחרוזת השנייה שהוכנסה, ומחריזה 1 אם כן ו – 0 אחרת בתוך כתובת הזבל שבמחסנית.

**swapAddress** – מקבלת כתובות של שני איברים ברשימה בתוך המחסנית ומחליפה את הכתובות של המחרוזות אליהן מצביעים האיברים.

**RevBubble** – שגרה רקורסיבית: מקבלת מילון ממוין פרט אולי לאיבר האחרון וממיינת אותו.  
מקבלת במחסנית את הערך של האיבר הנוכחי ברשימה (איבר התחלתי בקריאה ראשונה הוא האיבר הראשון במילון). תנאי העצירה הוא שהאיבר הבא הוא האיבר לפני האחרון ברשימה. קיימים גם תנאי עצירה קיצוניים נוספים (מקרי קצה): האיבר הבא הוא האיבר האחרון / ערך המילה באיבר הנוכחי הוא 0 (מילה ריקה). אם אף אחד מתנאי העצירה לא מתקיים – תוכנס למחסנית כתובת האיבר הבא ברשימה ותתבצע קריאה חוזרת לשגרה. אחרת, נתחיל בתהליך המיון: כשיתקיים תנאי העצירה, נהיה בעומק הרקורסיה, במצב בו האיבר הנוכחי אשר יושב במחסנית הוא האיבר לפני האחרון. תהליך המיון מתבצע באופן הבא: השגרה בודקת האם המילה היושבת באיבר הנוכחי גדולה לקסיקוגרפית מהמילה היושבת באיבר הבא בעזרת השגרה **isLexiBigger**. אם לא, תקפוץ לסוף: תרוקן את המחסנית מהאיבר הנוכחי ותסיים. אחרת, תבצע את שגרת **swapAddress** עבור האיבר הנוכחי והאיבר הבא. כעת, בקריאה הקודמת לקריאה העמוקה ביותר, האיבר הגדול ביותר ברשימה נמצא במקומו ונבצע החלפה, אם צריך, עם זוג האיברים הקודם לו. כך הלאה נחזור לאחור ברשימה עד שנגיע לאיבר הראשון מהקריאה הראשונה לשגרה, נבצע החלפה במקרה הצורך, ונסיים.

**clock** – **שגרת שירות פסיקת** **השעון** לאורך ריצת התוכנית. נכנס לשגרה זו הרבה במהלך התוכנית, ולכן שגרה זו חסכונית בזמן ריצתה: משתמשת באוגר משותף r5 המיועד לספירת תקתוקי השעון אשר עברו בשנייה הנוכחית. אם הגענו לאפס – נפחית את זמן ההודעה (היושב בכתובת משותפת msg\_time) באחד, ונאתחל את r5 לערך היושב בתווית rate אשר מייצג את מספר תקתוקי השעון בשנייה.

**insertToList** – מקבלת במחסנית את כתובת המילון ואת הכתובת בה נמצאת המילה אשר נרצה להוסיף למילון. השגרה רצה על המילון עד הגעה לאיבר הפנוי הראשון (איבר המכיל מחרוזת ריקה 0) ואז דורסת את ערך המחרוזת הריקה עם הערך הנמצא במילה שקיבלה במחסנית.

**concat** – מוסיפה את המילה הנוכחית להודעה (כתובות המחרוזות מועברות בצורת בכתובת משותפת) בכך שדורסת את ה string terminator '@' בהודעה עם התו ' ' (רווח) ואז מעבירה תו-תו את התוכן המילה הנוכחית להודעה ומוסיפה מחדש את '@' להודעה.

**invertString** – מקבלת כתובת של מחרוזת במחסנית אשר מסתיימת ב – '@', והופכת את תווי המחרוזת עד ההגעה ל – '@'. (לדוגמה, cat -> tac).

**toStringPrint** – מקבלת מספר במחסנית וממירה את כל הספרות בו לתווים על ידי ניצול העובדה שערכי ה ASCII של הספרות בין 0 ל – 7 הן 60 עד 67 בהתאמה, ולכן מוסיפה 60 לכל ספרה על מנת להמירה לערך ה ASCII של התו, ואז מחלקת את המספר ב – 10 (ASR 3 פעמים) על מנת להגיע לתו הבא. כך מורכבת המחרוזת של הספרות, ולאחר מכן השגרה קוראת ל **invertString** עם המחרוזת שהרכיבה על מנת לקבל את הערך המספרי הנכון (ולא ההפוך), ולבסוף שולחת את המחרוזת להדפסה בעזרת השגרה **printSentence**.

**printTime** – השגרה מדפיסה את ההודעה המציינת כמה זמן נותר למשתמש להקליד את המילים עבור הרצת המשחק ונעזרת ב **toStringPrint** עם הפרמטר של כמות הזמן שנותרה לתוכנית, על מנת להדפיס למשתמש את מספר זה. לבסוף מתבצעת ירידת שורה.

**msgPrint** - מדפיסה למשתמש את הודעות הפלט הקבועות המתאימות ואת ההודעה אשר כתב עד כה, ולבסוף נעזרת ב **printTime** על מנת להדפיס למשתמש כמה זמן נותר לריצה.

**clearWord** – מכינה את התוכנית לקליטת המילה הבאה: מאפסת buffer (כתובת משותפת המכילה את המילה שהתקבלה עד כה), ומאתחלת את המצביע ל buffer להימצא בתחילתו (איפוס המילה).

**addToDict** – משתמשת ב **insertToList** על מנת להוסיף את המילה הנוכחית היושבת ב buffer (כתובת משותפת) למילון. לאחר מכן, משתמשת ב **concat** על מנת לאחד את המילה היושבת ב buffer עם ההודעה עד כה. כעת נעזרת ב **msgPrint** על מנת להציג למשתמש את ההודעה עד כה ואת הזמן שנותר לתוכנית. צעד הבא: קריאה ל **clearWord** על מנת לאתחל את ה buffer ואת המצביע לכתובת הבית ההתחלתי שלו. מתבצע קידום הכתובת המשותפת dict\_len המציינת את אורך המילון, ולבסוף: אם נותרו למשתמש עוד מילים להקליד, מותירים למקלדת לבצע פסיקות נוספות, ואחרת לא מעלים את ה interrupt enable ו – write enable עבור התקן המקלדת.

**addNotDict** – מבצעת את אותן הפעולות כמו השגרה **addToDict** עם שינוי אחד: לא מתבצעת קריאה ל **insertToList.** כלומר – המילה לא מתווספת למילון.

**isPrefix** – שגרה המקבלת כפרמטרים במחסנית את כתובת זבל אליה תכתב התוצאה\*, כתובת של מילה, כתובת של תחילית פוטנציאלית למילה ומספר הטעויות המותר. מחזירה בתוך הכתובת המוקצית במחסנית 1 אם התחילית היא תחילית של המילה הנתונה עם מספר הטעויות המותר **בדיוק**\*\*, ו – 0 אחרת. השגרה הינה שגרת מעטפת לשגרה הרקורסיבית **auxIsPrefix**:

**auxIsPrefix**- שגרה רקורסיבית המוכלת בתוך **isPrefix**. השגרה מקבלת במחסנית כתובת של תו נוכחי במילה, כתובת של תו נוכחי בתחילית הפוטנציאלית ומספר טעויות מותר. בודקת התאמה עבור כל אחת מהאפשרויות לטעויות – התאמה מלאה (תחילית מושלמת), השמטה של תווים, הוספה של תווים או החלפה של תווים, בגדר מספר הטעויות המותר X ובכל קומבינציה אפשרית לטעויות.

תנאי העצירה הם:   
1. המצביע לתחילית מצביע לסוף התחילית. במצב זה נבדוק אם X הוא 0 ואם כן – נחזיר 1.   
2. מספר הטעויות קטן מ 0. במצב הזה התחילית היא לא תחילית של המילה.

אם שני התווים במילים זהים: יקודמו לתווים הבאים במחרוזות ותקרא קריאה מחודשת לשגרה עם התווים הבאים. אחרת, יש טעות: נוריד 1 מהכתובת במחסנית בה יושב מספר הטעויות X, וננסה לקרוא מחדש לשגרה עם כל אחד מהשינויים הבאים: קידום אות במילה אחת בלבד (שקול להוספה / החסרה של אות), או קידום של האותיות בשני המילים (שקול להחלפת אות). לאחר כל קריאה – נבדוק האם ערך ההחזרה הוא 1. אם כן: נגלגל אותו הלאה ונחזיר 1. כך בסופו של דבר יגיע ערך החזרה 1 מעומק הרקורסיה עד לקריאה הראשונית, והערך המוחזר הוא הערך הרצוי.

\*, \*\*: נעשו שינויים קטנים לשגרה **isPrefix** מהתרגיל הקודם. השינויים הם:  
1) ערך ההחזרה הוא במחסנית (ולא ברגיסטר r1),  
2) השגרה תחזיר 1 **אם ורק אם** התחילית היא תחילית של המילה עם **X טעויות בדיוק** (לא פחות)!

**flickerPrint** – מקבלת את הערכים הבאים במחסנית בסדר הבא: ערך זבל אשר יידרס עם תוצאת השגרה, אות המייצגת אי-בחירה של המשתמש, אות המייצגת בחירה של המשתמש ותו אשר יהבהב בהודעה. השגרה גורמת ללא הפסקה להבהוב בתו האחרון בהודעה (כפי שהוכנס במחסנית), בכך שמבצעת את רצף הדפסות התווים הבאים: backspace->space->backspace->char.  
לפני תחילת ההבהוב יוכנס 1 ל TKS על מנת לאפשר למשתמש להקליד תו (הדלקת ה write enable - צורת משאל). לפני כל תחילת רצף הבהוב, השגרה תבדוק האם התקבל תו מהמשתמש. אם התו הוא לא '!', תו הבחירה או תו אי-הבחירה שיושבים במחסנית אז השגרה תתעלם מתו זה, ותיתן למשתמש להקליד תו נוסף עד שיתקבל תו חוקי.  
אם התקבל תו הבחירה: תוצאת השגרה תהיה 1.  
אם התקבל תו אי-הבחירה: תוצאת השגרה תהיה 0.   
אם התקבל התו '!': תוצאת השגרה תהיה 1-.

**wordSuggestion** – מקבלת במחסנית ערך זבל, מספר טעויות, ומילה במילון אשר מילת המשתמש מהווה תחילית שלה לפי מספר הטעויות הנתון. השגרה תדפיס הודעה למשתמש ואז תקרא ל - **flickerPrint** על מנת לחכות לתשובת המשתמש (האם בחר במילה או לא). אם המילה היא תחילית מושלמת אז תווי הבחירה של המשתמש הם רווח ואנטר, אחרת הם 'y' ו – 'n' ותתווסף הודעה שונה למשתמש המיידעת אותו כי מדובר בתחילית לא מדויקת. ערך ההחזרה של **flickerPrint** מועבר במחסנית להיות ערך ההחזרה של **wordSuggestion**.

**suggestWords –** מציעה למשתמש את כל המילים במילון אשר מילת הקלט שהמשתמש הקליד מהווה תחילית שלהן עד כדי X טעויות, בסדר הבא: מ – 0 עד X טעויות ובסדר לקסיקוגרפי. מציאת התחיליות מתבצעת על ידי **isPrefix**. כאשר נמצאת תחילית, מתבצעת קריאה ל **wordSuggestion**. ערך ההחזרה של **wordSuggestion** נבדק: אם הוא 1 אז המשתמש בחר בתחילית. אם הוא 0 אז המשתמש לא בחר בתחילית והשגרה ממשיכה לחפש עבורו מילים במילון. אם הוא 1- אז המשתמש החליט לבטל את תהליך ההשלמה האוטומטית, ולכן נקפוץ לסוף. לאחר מעבר על כל המילים במילון X פעמים (פעם אחת עבור כל מילה עבור כל מספר טעויות מותר) – אם לא נמצאו מילים, השגרה תדפיס הודעה מתאימה. ההודעה תהיינה שונה אם הוצעו מילים והמשתמש לא בחר בהן לבין אם לא נמצאו כלל מילים למשתמש לצורך השלמה. ביציאה מהשגרה – אם לא נבחרה אף מילה, אז השגרה תדפיס מחדש את ההודעה שהמשתמש כתב, על מנת לאפשר לו להמשיך את המילה, ומאפשרת להתקן המקלדת לפסוק.

**msgScan** – **שגרת שירות המקלדת** לאורך ריצת התוכנית. השגרה קוראת את התו אשר יושב ב TKB ובודקת האם הוא אחד מהתווים המיוחדים: "space, backspace, . , !". עבור כל תו שאינו אחד מהתווים המיוחדים (או אנטר, אשר נחסם להדפסה וקליטה, ומתעלמים ממנו לגמרי), השגרה תדפיסו ותוסיף אותו ל buffer היושב בכתובת משותפת וקולט את המילה הנוכחית, ותקדם את המקום בזיכרון אשר מצביע לכתובת אליה הגענו בתו הנוכחי ב buffer עבור המילה הנוכחית. במקרה של backspace, תבדוק האם המשתמש מנסה למחוק את `> `, ואם כן – תחסום הדפסה וקליטה של תו זה ותאפשר קליטת התו הבא. אחרת יימחק התו הקודם מהמילה עד כה.  
במקרה של '.' , לא תתאפשר קליטה של תו נוסף (לא מעלה את writeEnable ב – TKS ל – 1), תדליק דגל משותף flag\_dot המודיע על קבלת תו העצירה, תדפיסו ותצא מהשגרה.  
במקרה של רווח או '!': התו שמוכנס ל buffer המכיל את המילה עד כה הוא '@' ובנוסף, עבור כל אחד מהתווים ' ', '!' תתבצע פעולה שונה:

במקרה של ' ' (**רווח**): תתבצע קריאה לשגרה **addToDict** האחראית להוספת המילה ב buffer למילון ולהודעה, ולאפס את ה buffer.

במקרה של '!': תתבצע קריאה לשגרה **suggestWords** האחראית על הצעת המילים למשתמש שהמילה הנוכחית אשר הקליד מהווה תחילית שלהן (עד כדי X טעויות), באופן שתואר קודם.

בסוף הקליטה, מחזירים למקלדת את האפשרות לקלוט תווים (ולפסוק), אלא אם התקבל רווח, ! או נקודה '.' (במקרים אלה, נעלה בתנאי את האפשרות לקלוט, פרט לקליטת נקודה אשר עוצרת את פסיקות המקלדת לעד). בנוסף, עבור כל אות שנוסיף למילה הנוכחית, נקדם מצביע היושב בכתובת משותפת ומראה באיזו כתובת בזיכרון נמצא התו הבא להכנסה לתוך ה buffer, לצורך כתיבה למקום הנכון קבלת הפסיקה בכל פעם.

**הסבר על מבני הנתונים בתוכנית:**

**רשימה:**  
כפי שצוין בתיאור השגרות, התוכנית משתמש במבנה נתונים של רשימה עבור שמירת המילים שבמילון בזיכרון. הרשימה כולה מתווספת לתוכנית לצורך הרצה (לא פריט מימוש שלנו), ותחת התווית **Dict** המציינת את האיבר הראשון ברשימה.  
מבנה הרשימה הוא כדלהלן: הרשימה מורכבת מאיברים, כאשר בכתובת של כל איבר יושבות שתי כתובות אשר משמשות כמצביעים לכתובות קיימות בזיכרון: הראשונה מכילה כתובת של מילה, והשנייה מכילה כתובת של האיבר הבא ברשימה. האיבר האחרון ברשימה מכיל את הערך 0 (כתובת NIL) כמצביע לאיבר הבא. נציין שוב כי המילון וכל הכתובות (והגודל המוקצה עבורן) מתווספות לתוכנית ואינן פריט מימוש. בנוסף, הנחה נוספת לגבי המילון היא שהמילים היושבות בו נמצאות בסדר לקסיקוגרפי, מהקטן לגדול.

**שגרות התוכנית אשר רשאיות לגשת לרשימה:**

**dictCalcLen** – לצורך קריאה בלבד: חישוב גודל הרשימה.  
**swapAddress** – לצורך כתיבה: החלפת הכתובות המייצגות מחרוזת היושבות בשני תאים.  
**revBubble** – לצורך קריאה וכתיבה: בדיקת נכונות הסדר הלקסיקוגרפי בין כל שני איברים ברשימה, ושימוש ב **swap** לצורכי כתיבה על מנת להחליף בין המחרוזות היושבות בין שני איברים לא ממוינים.  
**insertToList** – לצורך כתיבה: דריסה של המחרוזת הריקה הראשונה המופיעה במילון עם מחרוזת חדשה המייצגת מילה שנרצה להוסיף למילון.  
**suggestWords** – לצורך קריאה בלבד: עוברת על כל מילות הרשימה על מנת לנסות למצוא מילה שמילת הקלט שהמשתמש הקליד יכולה להוות תחילית שלהן, עד כדי X טעויות.