הטכניון - מכון טכנולוגי לישראל
הפקולטה להנדסת חשמל
מבוא למערכות תוכנה

תרגיל בית 3 - ADT	
מועד ההגשה: יום ד', 31/05/2018 בשעה 23:59	
האחראי על התרגיל: יואב רובין	yoavrubin17@gmail.com

ניהול כוחות בשדה קרב

בתרגיל זה נבנה שלד של מערכת לניהול כוחות בשדה הקרב.
אנחנו ננהל חיילים ונגמ"שים, נחלק אותם לחוליות שונות ואת החוליות נפזר בין אזורי הקרב השונים בשדה הקרב.

נשתמש בשני סוגים של טיפוסי נתונים ונבצע את החלקים הבאים:

1. טיפוס נתונים ספציפי.
   ניצור שני טיפוסי נתונים ספציפיים חדשים בהם נשתמש בתרגיל:
   חייל , נגמ"ש.
2. טיפוס נתונים מופשט ADT.
   ניצור רשימה מקושרת שתוכל לעבוד עם כל טיפוס נתונים שנבחר.
3. ניצור טיפוסי נתונים ספציפיים אשר ישתמשו ב- ADT שיצרנו בחלק 2.
4. ניצור קובץ main לעיבוד הקלט לתכנית.

קראו את חלק 5 "הערות לתרגיל כולו" לפני שאתם מתחילים לעבוד.

חלק 1 – יצירת טיפוסי נתונים ספציפיים

טיפוס 1 : Soldier (לא ADT)

תחילה נממש טיפוס עבור חייל.
טיפוס זה יוגדר וימומש בקבצים Soldier.h, Soldier.c.

עבור כל חייל נשמור:

1. מזהה (ID): מחרוזת מהצורה:SXXX כאשר X הוא תו כלשהוא.
2. תפקיד (Pos): אחת מהמחרוזות- DRIVER , MED , ENG , INT , GUNNER.

- שני חיילים זהים אם יש להם את אותו המזהה.
- אסור שיהיו שני חיילים זהים בתכנית.

על מנשק החייל לתמוך בפעולות הבאות:

1. Soldier_Create
   הפונקציה מקבלת מזהה ותפקיד ויוצרת חייל חדש בזיכרון עם פרטים אלה.
   הפונקציה תחזיר מצביע לחייל החדש שנוצר או NULL במידה ונכשלה.

2. Soldier_Delete
   הפונקציה מקבלת מצביע לחייל ומוחקת אותו מהזיכרון. (מחזירה void).

3. Soldier_Print
   הפונקציה מקבלת מצביע לחייל ומדפיסה את פרטיו באופן הבא: [ID] , [Pos]
   יש רווח בין כל פסיק לבין כל מילה וירידת שורה בסוף ההדפסה.
   לדוגמא: S123 , Med

4. Soldier_Duplicate
   הפונקציה מקבלת מצביע לחייל ויוצרת העתק חדש בזיכרון של החייל עם אותם הפרטים.
   הפונקציה תחזיר מצביע להעתק החייל החדש או NULL בכישלון.

- במידת הצורך, ניתן להוסיף פונקציות נוספות כרצונכם.

טיפוס 2 : APC (לא ADT)

כעת נממש טיפוס עבור נגמ"ש. **מבנה הנגמ"ש יתפקד כמחסנית.**
טיפוס זה יוגדר וימומש בקבצים APC.h, APC.c.

עבור כל נגמ"ש נשמור:

1. מזהה (ID): מחרוזת מהצורה: AXXX כאשר X הוא תו כלשהוא.
2. מערך של מצביעים לחיילים. בתרגיל אורך המערך יהיה קבוע 6 (define).

- שני נגמ"שים זהים אם יש להם את אותו המזהה.
- אסור שיהיו שני נגמ"שים זהים בתכנית.

על מנשק הנגמ"ש לתמוך בפעולות הבאות:

1. APC_Create
הפונקציה מקבלת מזהה ויוצרת נגמ"ש ריק חדש בזיכרון. מחזירה מצביע לנגמ"ש החדש שנוצר או NULL במקרה של כישלון.

2. APC_Delete
הפונקציה מקבלת מצביע לנגמ"ש ומוחקת אותו (ואת החיילים) מהזיכרון. (מחזירה void).

3. APC_Print
הפונקציה מקבלת מצביע לנגמ"ש ומדפיסה את פרטיו באופן הבא:
[ID] , Occupancy: [Count]/[Max_Count]
[Count] = מספר החיילים בנגמ"ש. [Max_Count] = מספר המקומות הכולל בנגמ"ש.
ואז תדפיס את החיילים בנגמ"ש לפי הסדר - החייל בראש המחסנית יודפס ראשון.
A122 , Occupancy: 2/6
Seat 1: S234 , Driver
Seat 2: S111 , Med
יש רווח בין [Occupancy:] לבין [2/9].
יש רווח בין [Seat i:] לבין שאר השורה.
יש רווח בין כל פסיק לבין כל מילה וירידת שורה בסוף כל שורה.

4. APC_Duplicate
הפונקציה מקבלת מצביע לנגמ"ש ויוצרת העתק חדש בזיכרון של הנגמ"ש והחיילים בתוכו עם אותם הפרטים. הפונקציה תחזיר מצביע להעתק הנגמ"ש החדש או NULL בכישלון.

5. APC_Insert_Soldier
הפונקציה מקבלת מצביע לנגמ"ש ומצביע לחייל. אם יש מקום היא מוסיפה את החייל לתוך הנגמ"ש (לא יוצרת העתק שלו) ומחזירה הצלחה. אחרת מחזירה Failure ומדפיסה: Error: APC is Full (ירידת שורה בסוף).

6. APC_Pop
הפונקציה מקבלת מצביע לנגמ"ש. היא מוציאה את החייל מהנגמ"ש.
(לא מוחקת אותו מהזיכרון). בהצלחה תחזיר מצביע לחייל שהוצא מהנגמ"ש.
אחרת תחזיר NULL ותדפיס: Error: APC is Empty (ירידת שורה בסוף).

- במידת הצורך, ניתן להוסיף פונקציות נוספות כרצונכם.

חלק 2 – יצירת טיפוסי נתונים מופשטים ADT

בחלק זה ניצור ADT - טיפוס נתונים מופשט שבעזרתו נוכל להתעסק עם טיפוסים שונים מבלי שנצטרך לייצר מבנה נתונים חדש בעבור כל טיפוס וטיפוס.

List

נממש ADT של רשימה מקושרת.
טיפוס זה יוגדר וימומש בקבצים List.h, List.c.

על מנשק הרשימה לתמוך בפעולות הבאות:

1. PList List_Create(…)
הפונקציה מקבלת פונקציות מחיקה, העתקה, השוואה והדפסה ובעזרתם יוצרת רשימה חדשה. בהצלחה מחזירה מצביע לרשימה החדשה שנוצרה או NULL בכישלון.

2. void List_Delete(…)
הפונקציה מקבלת מצביע לרשימה ומוחקת אותה מהזיכרון.

3. void List_Print(…)
הפונקציה מקבלת מצביע לרשימה ומדפיסה אותה.
ההדפסה צריכה להיות לפי סדר התאים מהראשון לאחרון.

4. Result List_Add_Elem(…)
הפונקציה מקבלת מצביע לרשימה ומצביע למידע. היא תיצור עותק חדש של המידע בזיכרון ותוסיף אותו לסוף הרשימה. מחזירה Success או Failure בכישלון.

5. Result List_Remove_Elem(...)
הפונקציה מקבלת מצביע לרשימה **ומפתח**. היא תמחק את המידע המתאים למפתח שקיבלה. אם לא קיים כזה תחזיר FAILURE אחרת תחזיר SUCCESS.

6. PElem List_Get_First(...)
הפונקציה מקבלת מצביע לרשימה ומחזירה את המידע הנמצא בתא הראשון ברשימה.

7. PElem List_Get_Next(...)
הפונקציה מקבלת מצביע לרשימה ומחזירה את המידע הנמצא בתא הבא ברשימה.
אם הגיעה לסוף הרשימה תחזיר NULL. (שימו לב שיש לזכור בכל פעם מה התא האחרון שהוחזר)

8. void List_Duplicate(...)
הפונקציה מקבלת מצביע לרשימת מקור ולרשימת יעד (ריקה) ומעתיקה את כל האיברים ברשימת המקור לרשימת היעד.

9. PElem List_Get_Elem(...)
הפונקציה מקבלת מצביע לרשימה **ומפתח**. הפונקציה תחזיר את המידע ברשימה המתאים למפתח או NULL אם לא נמצא כזה.

- במידת הצורך, ניתן להוסיף פונקציות נוספות כרצונכם.

<u>הערות לחלק זה</u>

1. ... = השלימו בעצמכם.
2. PList = מצביע למבנה ששמו: struct List.
3. בכדי לעבוד באמצעות הרשימה הנ"ל עם טיפוסים שונים – עליכם לספק פונקציות משתמש כפי שנלמד בכתה. אלה יהיו טיפוסי פונקציות המשתמש:
    4.1. CLONE_FUNC – מקבלת ומחזירה PElem.
    4.2. DESTROY_FUNC – מקבלת PElem ומחזירה void.
    4.3. COMPARE_KEYS_FUNC – מקבלת שני PKey ומחזירה bool.
    4.4. PRINT_FUNC – מקבלת PElem ומחזירה void.

שימו לב:
ההחלטה מה יהיה המפתח ואיפה הוא יישמר היא בחירה חופשית שלכם.
בעקבות החלטה זו הארגומנטים המתקבלים ע"י List_Add_Element יכולים להשתנות.
תלוי בכם ומותר לכם להוסיף ארגומנטים ולא להוריד.

חלק 3 – טיפוסי נתונים ספציפיים שישתמשו ב- ADT

טיפוס 1 : Squad

נממש מבנה עבור חולייה .
מבנה זה ימומש בקבצים Squad.h, Squad.c.

עבור כל חולייה נשמור:

1. מזהה (ID): מחרוזת מהצורה: SqXXX כאשר X הוא תו כלשהו.
2. מצביע לרשימה שתהיה רשימת החיילים בחולייה.
3. מצביע לרשימה שתהיה רשימת הנגמ"שים בחולייה.
4. מספר החיילים הכולל בחולייה: מספר שלם.

- שתי חוליות זהות אם יש להן את אותו המזהה.
- אסור שיהיו שתי חוליות זהות בתכנית.
- חייל בחולייה נמצא בתוך נגמ"ש (לא מופיע ברשימת החיילים) או מחוצה לו.

על מנשק החולייה לתמוך בפעולות הבאות:

1. Squad_Create
הפונקציה מקבלת מזהה ובנוסף זוגות של פונקציות מחיקה, העתקה, השוואה והדפסה ויוצרת חולייה חדשה בזיכרון. (כמו כן יוצרת את הרשימות הנחוצות). מחזירה מצביע לחולייה שנוצרה או NULL בכישלון.

2. Squad_Delete
הפונקציה מקבלת מצביע לחולייה ומוחקת אותה מהזיכרון.

3. Squad_Print
הפונקציה מקבלת מצביע לחולייה ומדפיסה את רשימת הנגמ"שים והחיילים שבה:

```
Squad: [ID] , Total troops: [Num]
APCs:
[ID] , Occupancy:  [Count]/[Max_Count]
[Soldiers inside APC by seats…]
[ID] , Occupancy:  [Count]/[Max_Count]
[Soldiers inside APC by seats…]
.
.
Soldiers:
[ID] , [Pos]
[ID] , [Pos]
```

[Num] = מספר החיילים בחולייה. ירידת שורה בסוף כל שורה.

4. Squad_Duplicate
הפונקציה מקבלת מצביע לחולייה ויוצרת עותק חדש שלה בזיכרון. מחזירה מצביע להעתק החדש שנוצר או NULL במקרה של כישלון.

5. Squad_Add_Soldier
הפונקציה מקבלת מצביע לחולייה, תפקיד ומזהה של חייל. אם ניתן, יוצרת חייל חדש בעל פרטים אלה ומוסיפה אותו לחולייה. מחזירה SUCCESS או FAILURE בכישלון.

6. Squad_Add_APC

הפונקציה מקבלת מצביע לחולייה ומזהה של נגמ"ש. אם ניתן, יוצרת נגמ"ש חדש בעל פרטים אלה ומוסיפה אותו לחולייה. מחזירה SUCCESS או FAILURE בכישלון.

7. Squad_Insert_Sold_APC
הפונקציה מקבלת מצביע לחולייה ומזהים של חייל ונגמ"ש. אם ניתן, מכניסה את החייל המתאים לתוך הנגמ"ש המתאים ומחזירה SUCCESS. אחרת מחזירה FAILURE.

8. Squad_APC_Pop
הפונקציה מקבלת מצביע לחולייה ומזהה של נגמ"ש. אם ניתן, מוציאה את החייל מראש המחסנית של הנגמ"ש ומוסיפה אותו לרשימת החיילים בחולייה.
מחזירה SUCCESS או FAILURE.

9. Squad_Delete_Soldier
הפונקציה מקבלת מצביע לחולייה ומזהה של חייל. אם ניתן, מוחקת את החייל מהחולייה.
מחזירה SUCCESS או FAILURE

הבהרה: את החייל ניתן למחוק רק אם הוא נמצא ברשימת החיילים של החולייה. אם הוא נמצא בתוך נגמ"ש אז אי אפשר למחוק אותו מהחולייה. הפונקציה הנ"ל לא תוציא חייל מתוך נגמ"ש ותמחק אותו אלא רק תמחק חיילים שנמצאים ברשימת החיילים בחולייה.

10. Squad_Delete_APC
הפונקציה מקבלת מצביע לחולייה ומזהה של נגמ"ש.
אם ניתן, מוחקת את הנגמ"ש מהחולייה.
מחזירה SUCCESS או FAILURE

- במידת הצורך, ניתן להוסיף פונקציות נוספות כרצונכם.

טיפוס 2 : WarZone

נממש מבנה עבור אזור קרב .
מבנה זה ימומש בקבצים WarZone.h, WarZone.c.

עבור כל אזור קרב נשמור:

1. מזהה (ID): מחרוזת מהצורה: WXXX כאשר X הוא תו כלשהו.
2. מצביע לרשימה שתהיה רשימת החוליות באזור הקרב.
3. רמת כוננות – 0 הכי נמוך 3 הכי גבוה.

- שני אזורי קרב זהים רק אם יש להם את אותו המזהה.
- אסור שיהיו שני אזורי קרב זהים בתכנית.

על מנשק אזור הקרב לתמוך בפעולות הבאות:

1. WarZone_Create
הפונקציה מקבלת מזהה ופונקציות מחיקה, העתקה, השוואה והדפסה ויוצרת אזור קרב חדש בזיכרון. (יוצרת את הרשימות הנחוצות).
רמת הכוננות תהיה אפס. מחזירה מצביע לאזור החדש שנוצר או NULL בכישלון.

2. WarZone_Delete
הפונקציה מקבלת מצביע לאזור לחימה ומוחקת אותו מהזיכרון.

3. WarZone_Print
הפונקציה מקבלת מצביע לאזור לחימה ומדפיסה את רשימת החוליות שבו:

```
WarZone: [ID] , Alert State: [Alert]

Squad: [ID] , Total troops: [Num]
APCs:
[ID] , Occupancy:  [Count]/[Max_Count]
[ID] , Occupancy:  [Count]/[Max_Count]

Soldiers:
[ID] , [Pos] , [Rank]
[ID] , [Pos] , [Rank].
```

[Alert] = רמת הכוננות באותו אזור קרב. ירידת שורה בסוף כל שורה.

4. WarZone_Duplicate
הפונקציה מקבלת מצביע לאזור קרב ויוצרת עותק חדש שלו בזיכרון. מחזירה מצביע להעתק החדש שנוצר או NULL במקרה של כישלון.

5. WarZone_Raise_Alert
הפונקציה מקבלת מצביע לאזור קרב ומעלה את רמת הכוננות בו. אם רמת הכוננות הגיעה ל- 3 אזי מופעל נוהל חירום (הסבר בהמשך) ומיד לאחר מכן רמת הכוננות יורדת ל- 0. מחזירה את רמת הכוננות החדשה באותו אזור.

- במידת הצורך, ניתן להוסיף פונקציות נוספות כרצונכם.

Battlefield (שדה קרב)

בתכנית קיים שדה קרב יחיד.
שדה הקרב יכיל אזורי קרב War Zones.
כל אזור קרב יכיל את החוליות המוצבות בו.
אין הגבלה על מספר אזורי הקרב בשדה הקרב.
אין הגבלה על מספר החוליות באזור קרב.

שדה הקרב יאותחל בפונקציה הראשית.

לרשותכם זוג קבצים נוספים: Battlefield.h Battlefield.c.

אתם יכולים לממש בהם כל פונקציה נוספת לה תזדקקו על מנת שהתכנית תעבוד כראוי.

חלק 4– פונקציית main:

עליכם לכתוב קובץ main.c שיבצע ניתוח לקלט הסטנדרטי (שורה בכל פעם).
את הפקודות לניהול שדה הקרב תקבלו כקלט לתכנית. כל פקודה תהיה שורת קלט.
כפי שתראו מיד, לכל פקודה יש מספר סידורי.
בשל המרחק ממרכז הפיקוד והפרעות קשר – הפקודות מגיעות בסדר אקראי.
רק כאשר מתקבלת הפקודה המיוחדת: Exe מבצעים את כל הפקודות הממתינות על פי מספרן הסידורי ולא על פי סדר הגעתן.

<u>קלט</u>
- כל פקודה (למעט שתי פקודות) מגיעה בפורמט: [Index] [Command].
- [index] מספר סידורי של הפקודה. מספר שלם כלשהוא. מתחיל מ-1.
- [Command] הפקודה עצמה.
- יש רווח בין [Index] לבין [Command].
- רק הפקודות Exit ו- Exe יגיעו ללא אינדקס, כלומר רק [Command] ומתבצעות מיידית.
- ניתן להניח שעד לקבלת פקודת Exe אינדקסי שאר הפקודות יהיו רציפים.
  כלומר אם קיימות פקודות עם אינדקסים 1 ו-4 אזי בהכרח קיימות פקודות עם אינדקסים 2,3 עד פקודה ה- Exe הבאה.

<u>הנחות קלט</u>
1. ניתן להניח שלא יתקבלו שתי פקודות בו זמנית בעלות מספר סידורי זהה.
2. <u>לא</u> ניתן להניח שמספר הפקודות עד לקבלת הפקודה Exe יהיה חסום.
3. ניתן להניח כי הקלט חוקי – כלומר יתקבל בפורמט הנכון, אך עלולות להתקבל פקודות שאינן מוכרות.
4. הניחו כי אורך שורת הקלט המקסימלי הינו 256 תווים.
5. תמיד תתקבל הפקודה Exit.
6. לא תהיה התנגשות בין מזהים של חולייה וחייל. לא יהיה מזהה של חייל שיתחיל ב- Sq.

<u>עבור כל הפקודות</u>
1. אם לא קיים שדה קרב להדפיס: Error: No Battlefield
2. אם לא קיים אזור לחימה עם מזהה כלשהו יודפס: Error: No Such War Zone
3. אם לא קיימת חולייה עם מזהה כלשהו יודפס: Error: No Such Squad
4. אם לא קיים נגמ"ש עם מזהה כלשהו יודפס: Error: No Such APC
5. אם לא קיים חייל עם מזהה כלשהו יודפס: Error: No Such Soldier
6. אם אחד מהם כבר קיים וזה מהווה שגיאה יודפס: Error: X Already Exists
כאשר X יכול להיות: War Zone , Squad , APC , Soldier.

שימו לב: אם לא נאמר אחרת בפקודה ספציפית, לא מדפיסים את כל הודעות השגיאה בכל פקודה. מדפיסים רק את השגיאה של המיכל הגדול ביותר:
שדה קרב>אזור קרב>חולייה>נגמ"ש>חייל.
כלומר אם לא קיימת חולייה כזו וגם לא קיים אזור קרב כזה יודפס:
Error: No Such War Zone כי אזור קרב הוא מיכל גדול יותר מחולייה.

7. אם הפקודה לא מוכרת להדפיס: Error: Illegal Command ולהמשיך לפקודה הבאה.
8. בכל הדפסה: רווח בין כל מילה/מספר. ירידת שורה בסוף כל שורה.

<u>פקודות אפשריות</u>

1. [Index] Create_B

- יצירת שדה הקרב. פקודה זו תתקבל פעם אחת בלבד במהלך התכנית.

2. Exe

- גורמת לביצוע כל הפקודות שהגיעו עד עכשיו על פי סדר האינדקסים שלהן.
- בתום ביצוע כל הפקודות הנוכחיות להדפיס:

**********All Commands Executed**********

(10 כוכביות בכל צד ללא רווחים עם המילים בקצוות) + 2 ירידות שורה.

- אם אין פקודות לביצוע יודפס: No Commands to Execute

3. [Index] Add_W [ID]

הוספת אזור קרב עם מזהה [ID] לשדה הקרב.

4. [Index] Del_W [ID]

מחיקת אזור קרב עם מזהה [ID] משדה הקרב. (ואת כל תכולתו).

5. [Index] R_W [ID]

העלאת כוננות לאזור קרב עם מזהה [ID].
אם אזור לחימה מגיע לרמת כוננות 3 אזי מוגדר באזור זה מצב חירום.

<u>מצב חירום</u>

במצב חירום כל היחידות בשדה הקרב כולו יועברו לאזור הלחימה הנ"ל ומצב הכוננות שלו ירד חזרה ל- 0.

6. [Index] Add_Sq [W_ID] [S_ID]

הוספת חולייה חדשה בעלת מזהה [S_ID] לאזור קרב בעל מזהה [W_ID]

7. [Index] Del_Sq [W_ID] [S_ID]

מחיקת חולייה בעלת מזהה [S_ID] מאזור לחימה בעל מזהה [W_ID]

8. [Index] M_Sq [Origin_W] [Dest_W] [S_ID]

העברת חולייה עם מזהה [S_ID] מאזור קרב עם מזהה [Origin_W]
לאזור קרב עם מזהה [Dest_W]
אם לא קיים אזור קרב (המקור) יודפס: Error: No Such Origin War Zone
אם לא קיים אזור קרב (היעד) יודפס: Error: No Such Dest War Zone
אם שניהם לא קיימים – שגיאת המקור תודפס בלבד.

9. [Index] Add_Sold [W_ID] [S_ID] [ID] [Pos]

הוספת חייל חדש בעל מזהה [ID] ותפקיד [Pos]
לחולייה עם מזהה [S_ID] באזור קרב עם מזהה [W_ID]

10. [Index] Del_ Sold [W_ID] [S_ID] [ID]

מחיקת חייל בעל מזהה [ID]
מחולייה בעלת מזהה [S_ID] באזור קרב עם מזהה [W_ID]

11. [Index] Add_APC [W_ID] [S_ID] [ID]
הוספת נגמ"ש חדש בעל מזהה [ID] לחולייה עם מזהה [S_ID]
באזור קרב עם מזהה [W_ID]

12. [Index] Del_APC [W_ID] [S_ID] [ID]
מחיקת נגמ"ש בעל מזהה [ID] מחולייה בעלת מזהה [S_ID]
באזור קרב עם מזהה [W_ID]

13. [Index] Sold_Insert [W_ID] [S_ID] [APC_ID] [Sold _ID]
הכנסת חייל בעל מזהה [Sold_ID] לתוך נגמ"ש בעל מזהה [APC_ID].
החייל והנגמ"ש יהיו מחולייה בעלת מזהה [S_ID] באזור קרב בעל מזהה [W_ID].
החייל יוסר מרשימת החיילים של היחידה ויועבר למערך הנגמ"ש.
אם הנגמ"ש מלא יודפס: Error: APC Is Full

14. [Index] APC_Pop [W_ID] [S_ID] [APC_ID]
הוצאת חייל מראש הנגמ"ש בעל המזהה [APC_ID]. החייל והנגמ"ש יהיו מחולייה בעלת מזהה [S_ID] באזור קרב בעל מזהה [W_ID]. החייל יוכנס לרשימת החיילים של היחידה ויימחק ממערך הנגמ"ש.
אם הנגמ"ש ריק יודפס: Error: APC Is Empty

15. Exit
יציאה מהתכנית ושיחרור כל הזיכרון.

16. [Index] Print
הדפסת הכותרת: Battlefield + ירידת שורה ואז הדפסת שדה הקרב כולו.

לעזרתכם נתונה דוגמה של תוכנית המבצעת ניתוח של הקלט תוך שימוש בפונקציות strtok, fgets ו- atoi:

```
int main() {
  char szLine[MAX_LINE_SIZE];
  char* delimiters = " \t\n";
  char* pszName;
  char* pszYear;
  int year;

  while (fgets(szLine, MAX_LINE_SIZE, stdin)) {
    printf("> %s", szLine);
    pszName = strtok(szLine, delimiters);
    pszYear = strtok(NULL, delimiters);
    if (pszName == NULL || pszYear == NULL)
    {
        fprintf(stderr, "Error: Not enough parameters\n");
    }
    year = atoi(pszYear);
    if (year == 0)
    {
        fprintf(stderr, "Error: Invalid year\n");
    }
    printf("%s is %d years old.\n", pszName, CURR_YEAR –
    year);
  }
  return 0;
}
```

תוכנית זו מצפה לקבל בכל שורת קלט שם ושנת לידה ומדפיסה את הגיל.

חלק 5 - הערות לתרגיל כולו

1. בתרגיל זה קיבלתם את כל הקבצים שיש להגיש. ברוב הקבצים כבר קיבלתם קטעי קוד מוכנים. אתם יכולים להשתמש בהם אך לא חייבים. אסור להוסיף קבצים נוספים משלכם לפרוייקט.

2. שימו לב כי נתון לכם קובץ defs.h המכיל את ההגדרה :
```
enum Result {FAILURE, SUCCESS};
```
במהלך התרגיל, עבור פונקציות המחזירות הצלחה/כשלון יש להשתמש בטיפוס זה.
בנוסף נתונים בקובץ זה קבועים נוספים לשימושכם.

3. בכל הדפסה: תמיד ירידת שורה בסוף כל שורה.

4. חל איסור להגדיר את המבנים השונים בקבצי ה- h. זה תכנות לא נכון ומפר מספר עקרונות שנלמדו בכתה. בנוסף שימו לב שאתם לא מערבבים משתנים פנימיים/שדות פנימיים/פונקציות פנימיות של טיפוס A בקבצי c של טיפוס B.

5. בדקו את הקוד ובצעו טסטים לכל חלק בנפרד! אם תכתבו את הכל בבת אחת יהיה לכם קשה לדבג ולתקן שגיאות זיכרון.

6. פונקציות C שימושיות לתרגיל:
strtok, strcmp, strlen, strcp , malloc, free, fgets.

7. ניתן להשתמש בדגל std=c99– בקובץ ה- makefile שלכם.

חלק 6 - תרגילון bash

כתוב סקריפט בשם spacesToDashes, בן 6 שורות לכל היותר (לא כולל השורה #!/bin/bash), שאינו מקבל פרמטרים, ואינו משתמש בקבצים זמניים.

על הסקריפט להחליף את כל הרווחים בקובץ הקלט שלו במקפים.
כל רווח יוחלף במקף בודד.
הסקריפט יקרא את הקובץ מערוץ הקלט הסטנדרטי.
כלומר, עבור קובץ הקלט "file1" הבא:
This is a test
והפקודה
spacesToDashes < file1
יודפס הפלט הבא:
This---is-a-test

טיפ- קראו באינטרנט על החלפת פרמטרים בbash בכתובת הבאה:
http://www.tldp.org/LDP/abs/html/parameter-substitution.html

-ניתן להניח שקובץ הקלט קיים ובנוסף שקיים בו לפחות רווח אחד.

הנחיות הגשה:

1. קבצי קוד חלקיים, וכן קבצי קלט ופלט לדוגמה, נמצאים בתיקייה
~eesoft/hmw/hmw3
לפני תחילת העבודה, הורידו את הקבצים לחשבונכם באמצעות הפקודה:
cp ~eesoft/hmw/hmw3/* .
2. עברו היטב על הוראות ההגשה של תרגילי הבית המופיעים באתר טרם ההגשה! ודאו כי התכנית שלכם עומדת בדרישות הבאות:
   - התכנית קריאה וברורה
   - התכנית מתועדת היטב לפי דרישות התיעוד המופיעות באתר
   - התכנית מתקמפלת ללא שגיאות וללא warnings כלל (יש להשתמש בדגל g –Wall)
   - התכנית רצה **ללא דליפות זיכרון** וגישות לא חוקיות לזיכרון כלל (בדיקה באמצעות valgrind)
   - התכנית נותנת פלט **זהה לחלוטין** לפלט הצפוי על כל קבצי הקלט שסופקו (בדיקה באמצעות פקודת diff על קבצי הפלט)
   - קובץ ה-makefile יוצר קובץ הרצה בשם הנדרש

3. יש להגיש קובץ tar יחיד המכיל את **כל הקבצים** שאתם נדרשים להגיש **ואותם בלבד** – ללא תתי-תיקיות. ודאו כי לא שכחתם את קובץ readme המכיל את פרטי הסטודנטים, וכן את ה-makefile במידה ונדרשתם.

4. שאלות בנוגע לתרגיל יש להפנות לפורום התרגיל ב-moodle בלבד

5. סיכום מפרט התרגיל:

סעיף	תיאור
נושא התרגיל	ADT
תאריך ההגשה	יום ד', 31/05/2018 בשעה 23:59
המתרגל האחראי על התרגיל	יואב רובין yoavrubin17@gmail.com
תיקייה המכילה קבצים לשימוש הסטודנטים	~eesoft/hmw/hmw3
קבצי הקוד הנתונים	parse_example.c, defs.h Soldier.h Soldier.c APC.h APC.c List.h List.c Squad.h Squad.c WarZone.h WarZone.c Battlefield.h Battlefield.c Main.c
קבצי הקלט והפלט הנתונים	in1.in out1.out in2.in out2.out
הקבצים שיש להגיש	readme makefile defs.h Soldier.h Soldier.c APC.h APC.c List.h List.c Squad.h Squad.c WarZone.h WarZone.c Battlefield.h Battlefield.c Main.c spacesToDashes

שם תכנית ההרצה הדרושה (הנוצרת ע"י makefile)	Battlefield
דגשים מיוחדים	

בהצלחה!