5. מחלקת מחסנית

התבניות שבמחלקה

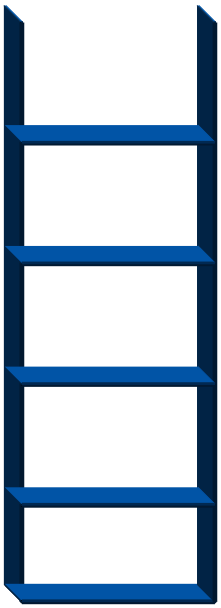
5.1 - הכנס נתונים למחסנית

5.2 - העבר למחסנית שניה

5.3 - הוצא נתונים ממחסנית

5.4 - הכנס למקום נבחר במחסנית

5.5 - איסוף בקיזוז



מחסנית היא מבנה נתונים אשר מתאים במיוחד למקרים בהם יש צורך לשמור סדרת נתונים ולהוציא אותם בסדר הפוך לסדר שמירתם. אחד המקרים הבסיסיים בהם יש צורך כזה הוא שמירת נתוני "ריצה" של תכנית רקורסיבית. במהלך זימון רקורסיבי נשמרים ערכי מהלך הביצוע; בזימון רקורסיבי נוסף נשמרים שוב ערכי מהלך הביצוע. עם החזרה מן הזימון השני, מוחזרים ערכי הביצוע שנשמרו בעת הזימון השני, הביצוע ממשיך מן הנקודה בה הופסק, ועם סיומו מוחזרים ערכי הביצוע שנשמרו בעת הזימון הראשון. בסך-הכל, נתונים של זימונים נשמרים עם "הירידה ברקורסיה", ומוחזרים עם "העלייה ממנה" בסדר הפוך לסדר שמירתם. מבנה הנתונים בו משתמש המחשב לשמירת הנתונים הוא מחסנית.

השימוש במחסנית מאפשר בעצם לא רק מימוש סדרה של זימונים רקורסיביים במחשב, כמתואר לעיל, אלא גם המרה של תכנית רקורסיבית לתכנית לא-רקורסיבית. אפשרות זו של המרה מדגישה את הרלוונטיות והאפקטיביות של מבנה הנתונים מחסנית. לא נרחיב בהמרה זו כאן כיון שהיא מעבר לחומר הלימוד הנדרש. מחסנית שימושית במרחב מגוון של בעיות לפתרון, וביניהן בעיות של סריקה/חיפוש במבני נתונים מורכבים כגון "עצים" (ראה בהמשך) או "גרפים", בעיות של זיהוי שייכות מילים לשפה (נושא אשר נלמד בפירוט ביחידת הלימוד "מודלים חישוביים"), בעיות שונות שבהן יש להפוך סדר של נתונים, ובעיות נוספות.

מבנה הנתונים מחסנית מאופיין בניהול נתונים מסוג LIFO (Last In First Out). כל הפעולות במבנה נתונים זה מתבצעות דרך "פתח כניסה ויציאה" אחד, הנקרא "ראש המחסנית". כלומר, נתון המוכנס למחסנית מושם בראשה, "מעל" כל הנתונים שבתוכה, ובעת שליפה מן המחסנית, מוצא הנתון שבראשה. אי-אפשר לשלוף נתון ממחסנית ריקה. ניתן לשלוף נתון ממחסנית גם כאשר אינו בראש המחסנית, אך לצורך כך יש תחילה להוציא את כל הנתונים שמעליו במחסנית, ואחר-כך להחזירם בסדר הפוך לסדר הוצאתם. לצורך כך נעשה שימוש במחסנית עזר, כפי שמוצג בתבניות שבפרק.

התבניות בפרק הן תבניות של הכנסת סדרות נתונים למחסנית, הוצאת סדרות נתונים מן המחסנית, הכנסת נתון למקום במחסנית שאיננו ראשה, או שימוש במחסנית לזיהוי סדרות נתונים מסוגים מסוימים. התבניות הראשונות, תבניות 5.1 עד 5.4, הן פעולות שכיחות בשימוש במחסנית, והן מופיעות בפיתוחי תכנה בהקשרים שונים. התבנית האחרונה, תבנית 5.5, כוללת שימוש במחסנית לזיהוי ועיבוד רצפים של נתונים. לפיכך, השאלות שאחרי התבניות 5.1 – 5.4 הן בעיקר שאלות של תרגול מגוון של הכנסה והוצאה ממחסנית, ואילו השאלות שאחרי תבנית 5.5 מתמקדות בזיהוי תכונות של רצפים של נתונים.

בפרט, השאלות שאחרי תבנית 5.5 מציגות בעיות כגון זיהוי סדרות של נתונים שבהן למשל, החצי השני של הסדרה הוא היפוך של החצי הראשון, או השלישים השני והשלישי של סדרה נתונה הם פונקציה של השליש הראשון. הפתרון של שאלות אלו מבוסס על "איסוף בקיזוז" באמצעות מחסנית, ובו תחילה מוכנסים נתוני קלט למחסנית ולאחר מכן הם מוצאים תוך כדי השוואתם לנתוני הקלט הבאים. לפעמים יש צורך ביותר ממחסנית אחת לביצוע החישוב הדרוש.

יחידות ספריה לטיפוס הנתונים **מחסנית** נמצאות באתר תבניות: www.tau.ac.il/~csedu

5.1 - הכנס נתונים למחסנית

נקודת מוצא: מחסנית מאותחלת, סדרת נתונים.

מטרה: הכנסת הנתונים למחסנית לפי סדר הגעתם.

אלגוריתם: **<u>הכנס-נתונים-למחסנית</u>** (S)

next_element ← איבר ראשון בסדרה

כל עוד לא סוף הסדרה בצע

פעולת-למחסנית (S, next_element)

next_element ← איבר הבא בסדרה

הערות

- תבנית זו היא תבנית בסיסית של הכנסת סדרת ערכים למחסנית. הנתון הראשון יהיה התחתון ביותר במחסנית מבין הנתונים המוכנסים. הנתון האחרון יהיה העליון ביותר, בראש המחסנית.
- דוגמה למצב בו נרצה לשמור במחסנית סדרת נתונים שהראשון בהם יהיה התחתון ביותר במחסנית והאחרון יהיה העליון ביותר היא הדוגמה הבאה. במבוא לפרק "רקורסיה" הצגנו בעיה של חישוב מספר מינימלי של פעולות מסוג 1+ ו- 2* הנדרש כדי להגיע ממספר חיובי X למספר חיובי Y, כאשר Y>X. נוח היה לחשב מספר זה באמצעות יישום גישה של חשיבה לאחור. לו היינו נדרשים לשמור את סדרת הפעולות, ולא רק לחשב את כמותן, היה מתאים להשתמש במחסנית. במהלך ההליכה לאחור מ-Y לכיוון X – כל פעולה (1+ או 2*) מחושבת ("אחורה") היתה מוכנסת למחסנית בעת חישובה. כלומר, תחילה היתה מוכנסת הפעולה האחרונה, שמביאה ל-Y, אחר-כך היתה מוכנסת הפעולה שמביאה למספר שלפני Y, וכך הלאה, ולבסוף היתה מוכנסת הפעולה הראשונה שיש לבצע, עם ההתחלה מ-X. (למשל, עבור זוג הנתונים 10 ו- 100, הייתה מוכנסת למחסנית תחילה הפעולה 2*, אחר-כך שוב 2*, אחר-כך 1+, וכך הלאה). כדי לקבל את סדרת הפעולות לפי הסדר שיש לבצען, על מנת להגיע מ-X ל-Y, תוצא סדרת הפעולות שהוכנסה בדיוק לפי הסדר שבמחסנית.

Pascal

```
(*  הפעולה מכניסה ערכים, מתוך סדרת נתונים, לתוך המחסנית S
                                    הנחה: S מחסנית מאותחלת.  *)
procedure stack_insert (var S: stack_type);
var
        next_element: stack_info_type;
begin
        get_element (next_element);
        while  not  end_of_data  do
          begin
                stack_push (S, next_element);
                get_element (next_element);
          end;
end:
```

C

```
/*  הפעולה מכניסה ערכים, מתוך סדרת נתונים, לתוך המחסנית S
                                    הנחה: S מחסנית מאותחלת.  */
void stack_insert (stack_type *S)
{
        stack_info_type next_element;

        next_element = get_element ();
        while (next_element != end_of_data)
         {
                stack_push (&S, next_element);
                next_element = get_element ();
         }

 }
```

5.2 - העבר למחסנית שניה

נקודת מוצא: שתי מחסניות מאותחלות, S1 ו- S2.

מטרה: העברת הנתונים מהמחסנית S1 למחסנית S2.

אלגוריתם: **<u>העבר-למחסנית-שניה</u>** (S1, S2)

כל עוד לא מחסנית-ריקה? (S1) בצע

הוצא-ממחסנית (S1), שלח-למחסנית (S2)

הערות

- לפעמים יש צורך לשמור את תוכנה, או חלק מתוכנה של מחסנית במקום נפרד. במקרים כאלה נסתייע במחסנית שניה, אליה יועבר תוכן המחסנית המקורית. דוגמה לכך תוארה במבוא לפרק, כאשר תוארה הפעולה של שליפת נתון שאיננו בראש המחסנית. בדוגמה זו הוזכר הצורך בשמירה מסודרת של הנתונים שמעל לנתון הנשלף, והחזרתם המסודרת לאחר השליפה. נוח לבצע שליפה מן המחסנית המקורית באמצעות מחסנית עזר, אליה יועברו הנתונים שמעל לנתון הנשלף.
- בתבנית המתוארת, מוצאים כל הנתונים מן המחסנית הראשונה, אך כמובן, ניתן לבצע את ההוצאה כל עוד מתקיים תנאי, למשל – כל עוד לא נשלף נתון מבוקש.
- סדר הנתונים במחסנית השניה יהיה הפוך לסדר שלהם במחסנית הראשונה. במידה ויוחזרו נתונים אלה למחסנית הראשונה, הם יסודרו בה לפי הסדר המקורי בו היו בה לפני ההעברה.

Pascal

```
(* הפעולה מעבירה את כל הערכים מהמחסנית S1 למחסנית S2
   הנחה: המחסניות מאותחלות. *)

procedure stack_transfer (var S1, S2: stack_type);
var
        x: stack_info_type;
begin
        while not stack_empty (S1) do
          begin
                stack_pop (S1, x);
                stack_push (S2, x);
           end;
end;
```

C

```
/* הפעולה מעבירה את כל הערכים מהמחסנית S1 למחסנית S2
   הנחה: המחסניות מאותחלות. */


void stack_transfer (stack_type *S1, stack_type *S2)
{

        while (! stack_empty (S1))
         {
                stack_push (S2, stack_pop (S1));
         }
 }
```

5.3 - הוצא נתונים ממחסנית

נקודת מוצא: מחסנית לא ריקה.

מטרה: הוצאת הנתונים מהמחסנית.

אלגוריתם: **<u>הוצא-נתונים-ממחסנית</u>** (S)

כל זמן ש**לא מחסנית-ריקה?** (S) בצע

שלוף-ממחסנית (S) ← element

הערות

- התבנית המוצגת היא תבנית בסיסית של הוצאת <u>כל</u> הנתונים שבמחסנית. וריאציה של תבנית זו היא הוצאה של חלק מן הנתונים בלבד. למשל, כל הנתונים העליונים, עד לנתון מסוים, או כל הנתונים שמתחת לנתון מסוים, או כל הנתונים שמקיימים תנאי מסוים.

- עבור המקרה האחרון המתואר בהערה הקודמת נשתמש במחסנית עזר. נתוני המחסנית המקורית יישלפו אחד-אחד וייבדקו. נתונים אשר עבורם יתקיים התנאי הנדרש יעובדו ולא יוחזרו למחסנית. נתונים אשר עבורם לא יתקיים התנאי הנדרש יוחזרו למחסנית המקורית, ויהיו מסודרים בה לפי הסדר המקורי בו היו שמורים בה מלכתחילה. לצורך כך, יהיה שימוש במחסנית עזר, אשר בה יישמרו הנתונים שיש להחזירם למחסנית המקורית, כפי שמוצג בפסאודו-קוד הבא. שים לב שהחזרת הנתונים ממחסנית העזר למחסנית המקורית מתבצעת תוך זימון התבנית 5.2, של "העבר למחסנית שניה".

אתחל-מחסנית ← S_help

כל זמן ש**לא מחסנית-ריקה?** (S) בצע

שלוף-ממחסנית (S) ← element

אם element **אינו מקיים תנאי** אזי

דחוף-למחסנית (S_help, element)

העבר-למחסנית-שניה (S_help, S)

- אפשרות נוספת של "הוצאה מותנית ואחריה החזרה" היא האפשרות של הוצאת כל הנתונים עד לנתון המקיים תנאי מסוים, שיש לשלפו ולעבדו, והחזרת כל הנתונים שהוצאו, כך שהסדר המקורי שלהם יישמר. אפשרות זו דומה מאד לפעולה המוצגת בתבנית הבאה, שבה מוכנס נתון במקום במחסנית אשר בו מתקיים תנאי דרוש.

Pascal

```
(*      הפעולה מוציאה את כל הערכים, מתוך המחסנית S
        הנחה: S מחסנית מאותחלת.      *)

procedure stack_clear (var S: stack_type);
var
        x : stack_info_type;
begin
        while not stack_empty (S) do
                stack_pop (S, x);
end;
```

C

```
/*   הפעולה מוציאה את כל הערכים, מתוך המחסנית S
        הנחה: S מחסנית מאותחלת.      */

void stack_clear (stack_type *S)
{

        while (! stack_empty (S))
         {
                stack_pop (S);
         }
}
```

שאלות

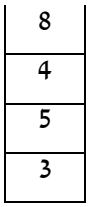
שאלה 5.3.1

כתוב אלגוריתם המקבל כקלט מחסנית ובה n איברים. על האלגוריתם להדפיס את ערכי האיברים לפי ההוראות הבאות:

א. מתחתית המחסנית אל ראש המחסנית.

ב. מראש המחסנית אל תחתית המחסנית.

דוגמה: אם המחסנית היא-



בסעיף א' יודפס: 8, 4, 5, 3

בסעיף ב' יודפס: 3, 5, 4, 8

■

שאלה 5.3.2

כתוב אלגוריתם המקבל כקלט מחסנית עם n איברים ממויינים בסדר עולה (האיבר הגדול נמצא בראש המחסנית). על האלגוריתם להוציא מהמחסנית את כל האיברים הגדולים או שווים ל- k.

■

שאלה 5.3.3

כתוב אלגוריתם המקבל כקלט מחסנית עם n איברים ממויינים בסדר עולה (האיבר הגדול נמצא בראש המחסנית). על האלגוריתם להוציא מהמחסנית את כל האיברים הקטנים או שווים ל- k.

■

שאלה 5.3.4 (מקור: מבנה נתונים, או"פ)

בחניון רכב יש שביל יחיד היכול להכיל עד 10 מכוניות. לשביל יש רק פתח אחד המשמש לכניסה ויציאה של מכונית, והוא נמצא בקצה השביל. כאשר לקוח מגיע לקחת את מכוניתו, והיא אינה הראשונה ליד היציאה, מוצאות כל המכוניות החוסמות לו את הדרך, לאחר מכן מוציאים את מכוניתו, ולבסוף מוחזרות המכוניות שהוצאו, כך שנשמר הסדר המקורי (מלבד המכונית שהוצאה).

פתח אלגוריתם שידמה את ניהול החניון. עבור כל מכונית המגיעה לחניון יתקבל כקלט התו A ומספר הרישוי של המכונית, ועבור כל מכונית המוצאת מהחניון התו D ומספר הרישוי של המכונית. כמו כן האלגוריתם יבצע את הפעולות הבאות:

א. יודיע על כל הגעה או עזיבה של מכונית.

ב. כשמגיעה מכונית חדשה יודיע אם יש או אין מקום פנוי בחניון. אם יש מקום פנוי תוכנס המכונית.

■

5.4 - הכנס למקום נבחר במחסנית

נקודת מוצא: מחסנית מאותחלת, איבר x.

מטרה: הכנסת נתון למקומו המתאים המחסנית.

אלגוריתם: **<u>הכנס-למקום-נבחר-במחסנית</u>** (S, x)

אתחל-מחסנית ← S_help

כל זמן לא מחסנית-ריקה?(S) *ו*בעת *הצץ-למחסנית*(S) > x *הכן*

פרוץ-למחסנית (*שלוף-ממחסנית* (S), S_ help)

פרוץ-למחסנית (S, x)

העבר-למחסנית-שניה (S_ help, S)

Pascal

```
(* הפעולה מכניסה איבר למחסנית, תוך שמירה על המחסנית ממויינת
   הנחה: האיבר הגדול נמצא בראש המחסנית *)

procedure insert_element (var S : stack_type; x : stack_info_type);
var
    S1 : stack_type;
    found : boolean;
    y : stack_info_type;
begin
    stack_init (S1);
    found := false;
    while (not stack_empty (S1)) and (not found) do
      begin
            if stack_top (S) > x then
              begin
                  stack_pop (S, y);
                  stack_push (S1, y);
              end
            else
                  found := true;
      end;
    stack_push (S, x);
    stack_transfer (S1, S);
end;
```

C

```
/* הפעולה מכניסה איבר למחסנית, תוך שמירה על המחסנית ממויינת
   הנחה: האיבר הגדול נמצא בראש המחסנית */

void insert_element(stack_type *S, stack_info_type x)
{
    stack_type S1;
    int found;

    stack_init (S1);
    found = 0;
    while (!stack_empty (&S1)) && (!found)

            if (stack_top(S) > x)
                  stack_push (&S1, stack_pop (S));
            else
                  found = 1;
    stack_push (S, x);
    stack_transfer (&S1, S);
}
```

שאלות

שאלה 5.4.1

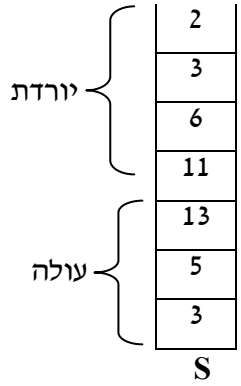
כתוב אלגוריתם המקבל כקלט מחסנית S עם n איברים. על האלגוריתם להחזיר מחסנית שניה S1 ובה נמצאים n האיברים שהיו ב- S רק בצורה ממויינת.

רמז: השתמש בתבנית 5.4 n פעמים (עבור כל איבר פעם אחת).

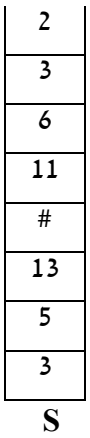
■

שאלה 5.4.2

מחסנית "עולה יורדת" היא מחסנית שהערכים מתחתית המחסנית עולים עד מקום מסוים (לאו דווקא אמצע המחסנית), ואז יורדים. לדוגמה S היא מחסנית עולה יורדת:



כתוב אלגוריתם שמקבל מחסנית "עולה יורדת" ומכניס את הסימן '#' בנקודה שבה המחסנית הופכת מעולה ליורדת. לדוגמה המחסנית S תראה כך:



■

5.5 - איסוף בקיזוז

נקודת מוצא: סדרה של ערכים. סדרה של נתונים.

מטרה: מאזן הערכים בסדרה, בהתאם לתנאים הנתונים.

אלגוריתם: **<u>איסוף בקיזוז</u>**

אתחל-מחסנית ← S

כל עוד לא סוף סדרת הקלט, בצע

next_element ← איבר הבא בסדרה

אם next_element מקיים תנאי-1 אזי

דחוף-למחסנית (S, next_element)

אחרת

אם מחסנית-ריקה? (S), אזי

החזר 'שקר'

אחרת

temp ← (S) שלוף-ממחסנית

אם temp אינו מקיים תנאי-2 אזי

החזר 'שקר'

החזר 'אמת'

הערות

- הרעיון בתבנית זו הוא עיבוד של סדרת נתונים שחלקם יישמרו וחלקם האחר יגרום להוצאה של הנתונים הנשמרים, ובסך-הכל יתקבל חישוב של מאזן שני סוגי נתונים שונים.

- למשל, כאשר נתון ביטוי חשבוני ובו וריאציות שונות של סוגריים, מסוגים שונים, רלוונטי לבדוק אם הסוגריים שבביטוי מסודרים בצורה חוקית. נתעלם לרגע מן התווים בביטוי שאינם פותחים או סוגרים. בביטוי הבא מסודרים הסוגריים בצורה חוקית: [[()]()] – לכל פותח ישנו סוגר מתאים בביטוי. לעומת זאת, בביטוי הבא הסוגריים אינם מסודרים בצורה חוקית: [(])()(], כיון שלפותח העגול הפנימי אין סוגר עגול מתאים. ניתן באמצעות שימוש במחסנית לבדוק חוקיות סוגריים, וזאת על-ידי מבט על סדרת הנתונים כמכילה תווים רלוונטיים משני סוגים – פותחים וסוגרים. הסוג הראשון – הפותחים – יהיה סוג של תווים שיישמרו במחסנית, והסוג השני – הסוגרים – יהיה סוג של תווים שיגרמו להוצאת תווים מן המחסנית. כלומר, כל פותח שייקרא יישמר במחסנית, וכל סוגר יוביל לשליפת הנתון שבראש המחסנית. כל עוד עבור כל סוגר יישלף מן המחסנית פותח מתאים, הרי הסוגריים חוקיים. בתבנית לעיל יהיה "תנאי-1" שוויון בין נתון הקלט האחרון לתו פותח, ו"תנאי-2" – התאמת "סוגר-פותח" בין נתון הקלט האחרון לנתון שבראש המחסנית. בדוגמה המסוימת הזו, של בדיקת חוקיות סוגריים יש גם להוסיף בסוף התבנית בדיקה שהמחסנית ריקה (כלומר, אין פותחים שחסרים עבורם סוגרים).

- התבנית מציגה מבנה סכמתי של חישוב או בדיקה כגון זה המודגם בהערה הקודמת. הבעיה הספציפית לפתרון יכולה כמובן להיות אחרת מזו המודגמת, וכך גם התנאים לבדיקה באלגוריתם. הרעיון המרכזי הוא שיש לשמור נתונים מסוימים (מסוג אחד), בסדר הפוך לסדר "הגעתם", ויש לשלוף נתונים נשמרים "בהגיע" נתונים נוספים (מסוג אחר). בזמן שליפה יש בדיקת התאמה בין נתון שגורם לשליפה לנתון נשלף. מבנה הנתונים המתאים לסוג כזה של עיבוד הוא מחסנית.

- אמנם בהערות לעיל מודגמים שני סוגי נתונים (פותחים וסוגרים), אך יתכן שיהיה רק סוג נתונים אחד, כאשר עד לנקודה מסוימת בסדרת נתוני הקלט (למשל, עד להופעת התו C) יחשבו כל הנתונים כנתונים שיש לשמרם, ומנקודה זו יחשבו כל הנתונים כנתונים שגורמים לשליפת נתונים שנשמרו. כך למשל, כאשר נרצה לבדוק האם סדרת תווים שבאמצעה מופיע התו C היא פלינדרום (הנחה: C אינו מופיע במקום אחר בסדרה). כאשר אין מגבלה על אורך הסדרה איננו יכולים להשתמש במערך לשמירת הנתונים. מבנה הנתונים המתאים הוא מחסנית, שבה יישמרו, עם קריאתם כל התווים שלפני C. כל תו הנקרא אחרי C יוביל לשליפת נתון מראש המחסנית והשוואתו לתו הנקרא. (במידה וההתקדמות תהיה מוצלחת, אזי בסוף התהליך תהיה גם בדיקה שהמחסנית ריקה.) בדוגמה זו, גם התווים הנשמרים וגם התווים שגורמים לשליפת התווים הנשמרים הם מאותו סוג. הגורם להבחנה בין שמירת תו נקרא לבין שליפה, בעקבות תו נקרא, הוא מקום הופעת התו הנקרא – לפני התו C או אחריו.

Pascal

```
(*  הפעולה מחזירה 'אמת' יש מאזן ערכים בסדרה בהתאם לתנאי
    ו-שקר' אחרת.                                        *)

function stack_gathering : boolean;
var
    S : stack_type;
    element, temp : stack_info_type;
    valid : boolean;
begin
    stack_init(S);
    element := get_element;
    valid := true;
    while (not end_of_input) and valid do
        begin
            if condition_1 (element) then
                stack_push (S, element)
            else
                if stack_empty (S) then
                        valid := false
                else
                    begin
                        stack_pop (S, temp);
                        if not condition_2 (temp) then
                            valid := false;
                    end;
            element := get_element;
        end;
    if not stack_empty (S) then
        valid := false;
    stack_gathering := valid;
end;
```

C

```
/* הפעולה מחזירה 'אמת' יש מאזן ערכים בסדרה בהתאם לתנאי
   ו-'שקר' אחרת. */

bool stack_gathering ()
{
    stack_type S;
    stack_info_type element, temp;
    stack_init(S);
    element = get_element();
    while (! end_of_input)
    {
        if condition_1 (element)
            stack_push (&S, element);
        else
            if stack_empty (&S)
                return 0;
            else
                {
                        temp = stack_pop (&S);
                        if (! condition_2 (temp))
                            return 0;
                }
        element = get_element();
    }
    if (! stack_empty (&S))
        return 0;
    return 1;
}
```

שאלות

שאלה 5.5.1

במסיבת ריקודים הוחלט שלריקוד הזוגות החדש יסתדרו המשתתפים באופן הבא: ראשית ייכנסו כל הנשים בזו אחר זו, ולאחר מכן ייכנסו הגברים ויצטרפו לנשים כך שהגבר הראשון שנכנס מצטרף לאחרונה שנכנסה, השני מצטרף לזו שלפני אחרונה וכן הלאה... הגבר האחרון שייכנס יצטרף לאישה שנכנסה ראשונה.

א. כתוב אלגוריתם שיבדוק האם הפרש הגילאים בין כל גבר ואישה שרוקדים יחד הוא הפרש סביר (נגדיר הפרש סביר כפחות מ- 10 שנים).

ב. מהי סיבוכיות האלגוריתם שכתבת כפונקציה של מספר המשתתפים במסיבה?

■

שאלה 5.5.2

במסגרת עיבוד ביטויים חשבוניים, עלה הצורך לבדוק את תקינות הסוגריים בביטויים המתקבלים. דמה את פעולת האלגוריתם לבדיקת תקינות סוגריים על המחרוזות הבאות ולגבי כל אחת מהמחרוזות שלהלן, הראה את תכולת המחסנית בכל שלב:

א. $\{ [a + b] - [(c - d)] \}$

ב. $(a + b) - \{ c + d \} - \{ f + g \}$

ג. $((h) * \{ ([j + k] \}))$

■

שאלה 5.5.3

פתח אלגוריתם שיקבע אם מחרוזת תווים היא מהצורה: $x\ C\ y$

כאשר x - היא מחרוזת תווים המורכבת מתווים A ו-B

ואילו y - היא היפוכה (תמונת הראי) של x.

דוגמא: אם $x =$ 'ABABBA' אז $y =$ 'ABBABA' והמחרוזת כולה היא:

ABABBA**C**ABBABA

בכל שלב באלגוריתם באפשרותך לקרוא רק את התו הבא במחרוזת.

מהי סיבוכיות האלגוריתם שכתבת כפונקציה של אורך הקלט?

■

שאלה 5.5.4

פתח אלגוריתם שיקלוט מחרוזת של תווים, תו אחר תו, ויקבע האם היא מהצורה:

$x_1 Z y_1 x_2 Z y_2 x_3 Z y_3 \ldots x_k Z y_k$ כאשר: $k > 0$

כל x_i - מחרוזת תווים מא"ב אנגלי, ללא האות Z

כל y_i - היפוך של המחרוזת x_i.

■

שאלה 5.5.5

א. פתח אלגוריתם שיבצע סימולציה של מעבד תמלילים פשוט, באופן הבא: יש לקלוט בזה אחר זה סדרה של תווים "רגילים" (המהווים את הטקסט) ותווים "מיוחדים" המהווים את הפעולות הבאות:

'.' (נקודה) - מציין סוף שורה. יש להדפיס את כל התווים שנקלטו לשורה זו.

'~' (טילדה)- מציין מחיקת תו. יש למחוק את התו האחרון שנקלט.

'*' (כוכבית) - מציין מחיקת שורה. יש למחוק את תווי כל השורה האחרונה.

תווים רגילים נקלטים ונשמרים לפי סדר הגעתם, ויודפסו רק כשיגיע הסימן "סוף-שורה".

ב. ממש את האלגוריתם בסביבת העבודה.

ג. מהי סיבוכיות האלגוריתם כפונקציה של גודל הקלט? נמק!

■

שאלה 5.5.6

כתוב אלגוריתם שיקלוט סדרת תווים בזה אחר זה, ומחזיר 'אמת' אם היא מהצורה הכללית: x%y%z, כך שהקלט מקיים את כל התנאים הבאים:

א. x היא מחרוזות המורכבת מהספרות 1,2,3 .

ב. y היא המחרוזת x בסדר הפוך, כאשר כל ספרה ב-x מופיעה פעמיים ב-y, ובין שני המופעים שלה מופיע $.

ג. z היא המחרוזת x בסדר המקורי, ובה כל ספרה מ-x מופיעה פעמיים רצופות.

דוגמא: מחרוזות תקינות: 123%3$32$21$1%112233 , 211%1$11$12$2%221111

מחרוזות לא תקינות: 34%4$43$3%3344 , 12%221$1%1212

האלגוריתם יחזיר 'שקר' אם סדרת הקלט אינה מקיימת את אחד מהתנאים הנ"ל.

מהי סיבוכיות האלגוריתם שכתבת, כפונקציה של גודל סדרת הקלט? נמק!

■

שאלות סיכום

שאלה 1

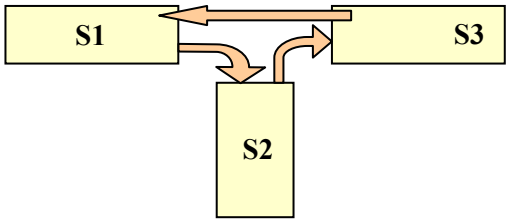
באולם כדורסל יש 3 מיכלים המכילים כדורים בשלושה צבעים: צהוב, כחול, אדום.
יש לסדר את הכדורים באופן הבא הצהובים במיכל 1
הכחולים במיכל 2
האדומים במיכל 3
פתח אלגוריתם שיקבל כפרמטר את שלושת המיכלים ויסדר את הכדורים לפי הסידור המבוקש. אין להשתמש במיכלים נוספים מלבד המיכלים הנתונים. הנח שיש מספיק מקום במיכלים.

■

שאלה 2 (מקור: שירלי רוזנברג-כהן)

נגדיר טיפוס נתונים חדש בשם **תלת-מחסנית**, כמבנה המכיל 3 מחסניות S1, S2, S3.

מבנה תלת-המחסנית נראה כך:



מעברים חוקיים ממחסנית למחסנית הם רק:
מ- S1 ל- S2, מ- S2 ל- S3, ומ- S3 ל- S1.

נגדיר את הפעולות הבאות על התלת-מחסנית:

העבר (Sx, Sy)	פעולה המעבירה איבר מראש מחסנית כלשהי Sx למחסנית כלשהי Sy. **הנחה:** Sx אינה ריקה.
האם-גדול? (Sx, Sy)	פעולה המחזירה 'אמת' אם האיבר בראש מחסנית Sx גדול או שווה לאיבר בראש מחסנית Sy, ו-'שקר' אחרת. **הנחות:** Sx ו – Sy לא ריקות.

נתונה תלת-מחסנית, שבה S1 מכילה מספרים שלמים לא ממויינים, S2 ו- S3 ריקות.

א. ייצג את הטיפוס **TStack**.

ב. כתוב אלגוריתם **מקסימום-בתלת-מחסנית (TS)**, המקבל תלת-מחסנית TS מסוג TStack, ומעביר את האיבר המקסימלי מ-S1 ל–S2. בסופו יהיו כל האיברים, למעט האיבר המקסימלי, ב- S1, ללא חשיבות הסדר.
השתמש בפעולות הממשק המוגדרות לעיל. אין להשתמש במחסניות נוספות.

ג. כתוב אלגוריתם יעיל **מיין (TS)**, המקבל תלת-מחסנית TS מסוג TStack, ומחזיר תלת-מחסנית ממויינת, כך שבאחת מהמחסניות נמצאים איברים ממויינים בסדר עולה (האיבר הגדול ביותר נמצא בתחתית המחסנית). השתמש באלגוריתם שכתבת בסעיף הקודם.

ד. מה סיבוכיות האלגוריתם שכתבת בסעיף ב', מתוך הנחה שבמחסנית S1 יש N איברים? נמק.

■

שאלה 3

נתון אלגוריתם:

```
אתחל-מחסנית S
קלוט מספר למשתנה y
דחוף-למחסנית (S, y)
קלוט מספר למשתנה y
כל זמן לא ריק (S) בצע
    הצץ-למחסנית (S) ← x
    אם x < y אזי
        כל זמן לא מחסנית-ריקה? (S) בצע
            שלוף-ממחסנית (S) ← temp
            דחוף-למחסנית (S, y)
        אחרת
        אם x = y אזי:
            דחוף-למחסנית (S, y)
            קלוט מספר למשתנה y
        כל זמן לא מחסנית-ריקה? (S) בצע
            שלוף-ממחסנית (S) ← x
        הדפס x
```

נתונים הקלטים הבאים: (משמאל לימין)

א. 2 , 2 , 1 , 4 ,1 , 3 , 4 , 2

ב. 3 ,5 ,13 ,10

1. עקוב אחר ביצוע האלגוריתם עבור כל קבוצת נתונים בנפרד והראה את מצב המחסנית בכל שלב.
2. מה מבצע האלגוריתם באופן כללי?
3. מה סיבוכיות האלגוריתם?
4. כתוב אלגוריתם יעיל יותר שמבצע את אותה פעולה.

■

שאלה 4

כתוב אלגוריתם מילולי המקבל מחסניות: S1 S2 ומחזיר 'אמת' אם שתי המחסניות זהות לחלוטין או שאחת הינה היפוך של השנייה. למשל עבור הדוגמאות הבאות יוחזר 'אמת'.

S1	S2
7	5
5	1
2	2
1	5
7	5

S1	S2
7	5
5	5
2	2
1	1
5	5

עבור הדוגמאות הבאות יוחזר 'שקר'.

S1	S2
5	
1	5
2	2
5	1
5	5

S1	S2
5	5
5	5
99	2
1	1
7	7

■

שאלה 5

נתון **חידה** אלגוריתמים הפועל על מערך של מחסניות A. האלגוריתם משתמש באלגוריתמים: **הוצא** ו**בדוק**. מהי מטרת האלגוריתם **חידה**? נמק.

<u>הוצא</u> (A)

עבור i *מ-* 1 *עד* N *בצע:*

שלוף-מאחסנית(A[i]) ← X

החזר A

<u>בדוק</u> (A)

עבור i *מ-* 1 *עד* N *בצע:*

אם מחסנית-ריקה? (A[i]) *אזי:*

החזר i

החזר 0.

<u>חידה</u> (A)

בדוק (A) ← F

כל זמן F = 0 *בצע*

הוצא (A) ← A

בדוק (A) ← F

החזר F

■