האוניברסיטה העברית בירושלים
בית הספר להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין

סדנת תכנות C ו-++C - תרגיל מסכם בשפת C (חלק ב׳)

מועד הגשה **(חלק ב׳):** יום ד' 24 למאי ב-22:00

נושאי התרגיל: #מצביעים #מצביעים_לפונקציות #תכנות_גנרי #ניהול_זיכרון

חלק ב - תכנות גנרי

בחלק זה של התרגיל נהפוך את הקוד שכתבנו בחלק א' לקוד גנרי.
מומלץ להשתמש בקוד שכתבתם לחלק א׳ ולעדכן אותו בהתאם לשינויים ולתוספות.

מי שלא בטוח מה זה קוד גנרי ואיך מממשים קוד כזה - נמליץ לו שיחזור על השיעורים והתרגולים לפני שהוא צולל לעומק התרגיל.

בחלק זה של התרגיל נעדכן את הספרייה markov_chain כך שנוכל ליצור שרשראות של טיפוסים שונים (ולא רק tweets / שרשראות של מחרוזות). כדי שנוכל לבחון את הספרייה הגנרית, תכתבו שני קבצים שהולכים להשתמש בספרייה.
הקבצים הינם: tweet_generator (בדומה למה שכתבתם בחלק א') ו-snakes_and_ladders (קובץ חדש).

שני הקבצים tweet_generator.c ו-snakes_and_ladders.c יכילו פונקציית main ונריץ כל פעם רק אחת מהן.

כדי לוודא שהספרייה markov_chain ממומשת באופן גנרי לחלוטין, הטסטים האוטומטיים של בית הספר ירוצו גם על **סטרקטים (סטרקט ברבים) שאתם לא מכירים**.

1 קבצים

כמו בחלק א׳, סיפקנו עבורכם קבצי קוד וקובץ קלט:

- justdoit_tweets.txt - קובץ הקלט של התכנית tweets_generator.
- markov_chain.h המכיל את השלד של הסטראקטים **המעודכנים.**
- linked_List.c linked_List.h - רשימה מקושרת לשימושכם.
- snakes_and_ladders.c - קובץ (ממומש חלקית) המשתמש בספריית מרקוב שכתבתם.

אתם צריכים להגיש:

- markov_chain.h מעודכן עם הסטראקטים אשר תכתבו.
- markov_chain.c מימוש של הפונקציות אשר נמצאות ב- markov_chain.h.
- tweets_generator.c דומה לקובץ אשר הוגש בחלק א׳, אך עם שינויים מותאמים למבנה הנתונים החדש.
- snakes_and_ladders.c הקובץ משתמש בספריית מרקוב שכתבתם, תצטרכו לממש אותו בהתאם להוראות שיפורטו בהמשך.
- makefile עם פקודות מתאימות קימפול והרצה של שתי תכניות שונות, אחת שמריצה את tweets generator ואחת את snakes and ladders שניהם רצות בעזרת ספריית מרקוב (פירוט בהמשך).

2 מבני נתונים

עליכם לשנות את ה-structs הבאים בקבצים markov_chain.h ו-markov_chain.c:

MarkovNode

- המצביע למילה יהפוך להיות מצביע לדאטה **גנרי**.

MarkovNodeFrequency

- .

MarkovChain

יש להוסיף **מצביעים לפונקציות גנריות(לא לממש אותם)**:

- print_func מצביע לפונקציה המקבלת מצביע מטיפוס גנרי, לא מחזירה כלום, ומדפיסה את הדאטה.
- comp_func מצביע לפונקציה המקבלת שני מצביעים לדאטה גנרי מאותו טיפוס, ומחזירה:
  - ערך חיובי אם הראשון יותר גדול מהשני;
  - ערך שלילי אם השני יותר גדול;
  - 0 אם שניהם שווים.
- free_data מצביע לפונקציה המקבלת מצביע מטיפוס גנרי, לא מחזירה כלום, ומשחררת את הזיכרון של המשתנה אותו קיבלה.
- copy_func מצביע לפונקציה המקבלת מצביע מטיפוס גנרי, ומחזירה העתק שלו המוקצה דינמית.
- is_last מצביע לפונקציה שמקבלת מצביע מטיפוס גנרי ומחזירה ערך בוליאני true אם הדאטה הוא האחרון בשרשרת מרקוב, ו-false אחרת.

<u>הנחיות:</u>

1. **מומלץ להשתמש ב-**typedef **כדי ליצור טיפוס חדש של מצביע לפונקציה, למשל:**

```
typedef bool (*is_even)(int);
```

מייצר טיפוס חדש בשם is_even של מצביע לפונקציה אשר מקבלת int ומחזירה bool.

2. בקובץ markov_chain.h יש את השלד של הסטראקט, **אסור** לשנות את השמות של המשתנים **ואסור** להוסיף עוד משתנים ל-Markov_Chain (כדי שנוכל להריץ טסטים).
3. בקבצים של ספריית מרקוב **<u>לא ממשים</u>** את הפונקציות הנ"ל, אלא רק **<u>מוסיפים מצביעים לפונקציות אלה</u>**. לכל אובייקט מסוג Markov_Chain יהיו פונקציות שמתאימות ספציפית לסוג המידע.
4. צריך להתאים את הקוד ב-markov_chain.c להשתמש ב**מצביעים של הפונקציות** לעיל במקום להשתמש בפונקציות של מחרוזות (למשל להשתמש ב-comp_func במקום ב-strcmp).

3 שימוש בספריית markov_chain על ידי tweets_generator

- הקלט והפלט זהים לקלט ופלט מחלק א' של התרגיל.
- שלב הלמידה ושלב יצירת הציוצים זהים מבחינה לוגית לשלבים אלו בחלק א', כלומר הלוגיקה נשארת זהה אך הקוד צריך להתאים לעדכונים בספריית markov_chain.
- כדי להתאים את הקוד ב- tweets_generator.c כך שיעבוד עם מבנה נתונים מעודכן, צריך לחשוב אילו פונקציות לספק ל-Markov_Chain בקובץ tweets_generator.c כדי שזה יעבוד על מחרוזות (טיפ: חלק מהפונקציות קיימות ב- <string.h>).
- מומלץ להשתמש בקובץ justdoit_tweets.txt כדי להריץ את התוכנית לוודא נכונות.

4 שימוש בספריית markov_chain על ידי snakes_and_ladders

כדי לבדוק גנריות, נרצה לבדוק את ספריית מרקוב על טיפוס נוסף, אנו נשתמש במשחק סולמות ונחשים לבדיקת ה-MakrovChain.

הנחיות והנחות:

- הלוח בגודל 100 (10*10), מתחיל מתא מספר 1, ומסתיים בתא מספר 100.
- לצורך פשטות, נניח שמשחקים את המשחק עם שחקן יחיד.
- **אנו נתייחס לכל משחק של שחקן יחיד כ"מסלול"** (כמו "משפט" בחלק א'). כל מסלול יכלול רצף חוקי של תאים.
- התוכנית תייצר מסלולים אפשריים של שחקן במשחק, מהתא הראשון בלוח (תא 1) לתא האחרון (100).
- ניתן להניח שאף תא לא מכיל נחש וסולם בו זמנית.

מסופק עבורכם קובץ snakes_and_ladders.c המכיל את הסטראקט הבא המייצג תא במשחק:

```
def struct Cell {
 number; // cell number (1-100)
 ladder_to;  // ladder_to represents the jump of the ladder in case there is one from this cell
 snake_to;   // snake_to represents the jump of the snake in case there is one from this cell
oth ladder_to and snake_to should be -1 if the cell doesn't have them
};
```

התכנית:

- קלט:

ערך **seed** – מספר שיינתן לפונקציית ה-srand() פעם אחת בתחילת ריצת התוכנית. ניתן להניח כי הוא מספר שלם אי שלילי (unsigned int).
כמות המסלולים שנרצה לייצר – ניתן להניח כי הפרמטר הוא מספר שלם וגדול ממש מ-0 (int).

בניגוד למייצר הציוצים, התוכנית לא מקבלת נתיב לקובץ קלט אלא מקבלת רק seed ומספר מסלולים, למשל הפקודה:

```
snakes_and_ladders 3 2
```

תריץ את התוכנית עם הערך 3 ל-seed ותדפיס שני מסלולים אפשריים של משחק.

- שלב הלמידה
  - **השלב כולו מומש עבורכם.** ודאו שאתם מבינים אותו וקוראים לפונקציות עם פרמטרים נכונים.
  - מימשנו עבורכם בקובץ snakes_and_ladders.c את הפונקציות create_board ו-fill_database. יש לעיין בקוד ולהבין אותו.
  - הפונקציה create_board משתמשת במערך הדו-מימדי transitions כדי לייצר את הלוח. השימוש בה מחליף את הקריאה מקובץ שקיימת במחולל הציוצים.
  - מימשנו עבורכם את fill_database כך שהמעבר בין שני תאים מוגדר עם הלוגיקה הבאה:
    - אם נמצאים בתא המכיל סולם אז תמיד "עולים" בסולם לתא שנמצא ב-ladder_to.
    - אם נמצאים בתא המכיל נחש אז תמיד "יורדים" לתא בקצה הנחש שנמצא ב-snake_to.
    - אחרת, נרצה לדמות זריקת קובייה, לכן מכל תא יש אפשרות לקפוץ לאחד משש התאים העוקבים באותה הסתברות. למשל: אם נמצאים כרגע בריבוע 50, ניתן לקפוץ לאחד התאים מ-51 עד 56 באותה הסתברות(בדומה להטלת קוביה).

- יצירת מסלול
  - בחירת התא הראשון במסלול: התא הראשון במסלול תמיד יהיה התא הראשון בלוח (ואין צורך לבחור את אחד מהתאים רנדומלית).
  - בחירת התא הבא: כמו בחלק א', נשתמש ב-database שיצרנו, ונבחר תא באופן רנדומלי מהתאים העוקבים של התא האחרון שבחרנו, כך שהסיכוי של כל תא עוקב להיבחר פרופורציונלי לתדירות שבה הוא מופיע.

  - מסלול מסתיים כשמגיעים לתא מספר 100 **או** לאחר ששיחקנו 60 סיבובים. כלומר:
    - תא 100 הוא "סוף משפט".
    - ו-60 הוא max_length (מספר המילים המקסימלי בכל ציוץ).

- פלט התוכנית תדפיס את המסלולים בפורמט הבא (הצבעים לא מודפסים, זה לנוחות קריאה):

Random Walk 1: [1] -> [5] -> [9] -> [11] -> [17] -> [23]-ladder to 76 -> [76] -> [77] -> [82] -> [84] -> [86] -> [92] -> [95]-snake to 67 -> [67] -> [68] -> [72] -> [75] -> [77] -> [81]-snake to 43 -> [43] -> [44] -> [47] -> [49] -> [52] -> [54] -> [59] -> [62] -> [66]-ladder to 89 -> [89] -> [95]-snake to 67 -> [67] -> [69]-snake to 32 -> [32] -> [33]-ladder to 70 -> [70] -> [75] -> [79]-ladder to 99 -> [99] -> [100]
Random Walk 2: [1] -> [3] -> [6] -> [10] -> [15]-ladder to 47 -> [47] -> [52] -> [55] -> [59] -> [65] -> [67] -> [68] -> [71] -> [73] -> [76] -> [77] -> [78] -> [84] -> [85]-snake to 17 -> [17] -> [22] -> [27] -> [31] -> [36] -> [42] -> [45] -> [46] -> [48] -> [51] -> [55] -> [56] -> [61]-snake to 14 -> [14] -> [20]-ladder to 39 -> [39] -> [43] -> [45] -> [48] -> [54] -> [56] -> [57]-ladder to 83 -> [83] -> [85]-snake to 17 -> [17] -> [18] -> [22] -> [24] -> [26] -> [29] -> [31] -> [36] -> [38] -> [43] -> [44] -> [48] -> [51] -> [57]-ladder to 83 -> [83] -> [88] -> [94] ->

פירוט פורמט ההדפסה:

1. כל מסלול צריך להסתיים בירידת שורה.
2. כל מסלול מתחיל בטקסט "Random Walk", אחריו מספר המסלול (מ-1 ועד כמות המסלולים) ונקודתיים - Random Walk i:
3. כל תא שעוברים בו ייכתב בתוך סוגריים מרובעים.
4. מעברים:
a. כל מעבר בין תאים ייכתב עם חץ בין התאים (ורווח יחיד בין שני צידי החץ).
b. כאשר יש **סולם** בין תא x לתא y נדפיס:
[x]-ladder to y-> [y]

c. כאשר יש **נחש** בין תא x לתא y נדפיס:
[x]-snake to y-> [y]

5. אם הגענו לריבוע ה-100 (ניצחון) המסלול מסתיים ב- [100].
6. אם הסתיימו 60 שלבים ולא הגענו ל-100 (כישלון) משאירים את החץ אחרי הריבוע האחרון.

קובץ snakes_and_ladders.c:

- עליכם לכתוב פונקציית main שמקבלת את הארגומנטים מה-CLI ומשתמשת בפונקציות הממומשות כדי ליצור ולמלא את ה-Markov_Chain ואז ליצור ולהדפיס מסלולים אפשריים לפלט.
- שימו לב, עליכם לכתוב ולספק ל-Markov_Chain מצביעים לפונקציות המתאימות לטיפוס Cell החדש, ממשו אותן בהתאם לצורך ובהתאם להוראות.

5 התאמות נוספות

- על מנת שהטסטים יעבדו וירוצו בצורה טובה על שתי התכניות, כל פונקציה שאתם ממשים ב-tweets_generator.c
ו- snakes_and_ladders.c **צריכה להיות סטטית.** (יש להוסיף את המילה static לפני השם של הפונקציה)

- החתימות של הפונקציות אשר מקבלות *char בקובץ markov_chain.h צריכות להשתנות בהתאם לשינויים שגוררים המעבר לתכנות קוד גנרי, חישבו כיצד.

- החתימה של הפונקציה add_node_to_frequencies_list בקובץ markov_chain.h השתנתה ל:

```
Bool add_node_to_frequencies_list (MarkovNode *first_node, MarkovNode *second_node,
Markov_Chain *markov_chain)
```

6 קובץ makefile

הסבר כללי:

בחלק זה של התרגיל נתרגל שימוש בסיסי ב-Make.
Make היא תוכנה לניהול אוטומטי של קומפילציית קוד, והיא חלק מפרויקט התוכנה החופשית GNU. כדי להשתמש ב-Make ניצור קובץ טקסט בשם makefile בו יכתבו ההוראות לקומפילציה, כאשר הפורמט הבסיסי להוראה הינו:

```
target_name: dependencies
commands
```

כאשר name_target הוא שם כלשהו (לבחירתכם), במקום dependencies נשים את קובץ הקוד שנרצה לקמפל, או שם של target נוסף שעבורו גם מוגדרות הוראות קומפילציה, ואת commands נחליף בפקודת הקומפילציה (אותה פקודה שהיינו כותבים בטרמינל).

לאחר שהגדרנו את הוראות הקומפילציה עבור target_name, נוכל להריץ בטרמינל את הפקודה make target_name , ו-make יריץ את commands והקומפילציה תתבצע.

בתרגיל זה:

בתרגיל זה עליכם להגיש makefile שיכיל שני target שונים, אחד לציוצים ואחד לסולמות ונחשים:

1. כאשר נריץ את הפקודה make tweets בטרמינל (בתיקיה עם קבצי הקוד), יווצר קובץ מקומפל אותו נוכל להריץ, למשל כך: ./tweets_generator 123 2 "justdoit_tweets.txt"
2. הפקודה make snakes תייצר קובץ מקומפל שניתן להריץ: ./snakes_and_ladders 3 2

מומלץ ליצור את הקובץ make **עם ה-**targets **הנ"ל בתחילת העבודה על התרגיל**, כך תוכלו להשתמש בפקודות הנ"ל בשביל לקמפל בקלות את הקבצים השונים בהתאם לתוכנית אותה אתם מעוניינים להריץ.

7 פתרון בית-ספר ו-presubmit

את בדיקת ה-presubmit תוכלו להריץ באמצעות הפקודה הבאה ב-CLI:

```
~labcc2/presubmit/ex3b/run
```

תוכלו להריץ את פתרון בי"ס במחשבי האוניברסיטה, או בגישה מרחוק בעזרת הפקודה הבאה ב-CLI:

```
<labcc2/school_solution/ex3b/schoolSolution <prog> <arguments~
```

שימו לב! בגלל שבחלק הזה יש שתי תוכניות צריך להגדיר ל-school_solution איזו מהן להריץ, כך ש- <prog> יכיל tweets אם נרצה להריץ את tweets_generator או יכיל את snakes אם נרצה להריץ את הסולמות והנחשים. את שאר הארגומנטים מוסיפים אחרי כרגיל, למשל:

```
~labcc2/school_solution/ex3b/schoolSolution snakes 3 2
```

הערה: הרנדומליות שונה ממחשב למחשב אפילו אם מקבעים את ה-seed עם srand. כדי להשוות עם פתרון בית הספר צריך להריץ על מחשבי האוניברסיטה כדי לקבל רנדומליות זהה.

8 דגשים והנחיות לתרגיל

- בסיום הריצה עליכם לשחרר את כלל המשאבים בהם השתמשתם, התוכנית שלכם תיבדק ע"י valgrind ויורדו נקודות במקרה של דליפות זיכרון.
- במקרה של שגיאת הקצאת זיכרון שנגרמה עקב calloc()/realloc()/malloc(), יש להדפיס הודעת שגיאה מתאימה ל-stdout המתחילה ב- "Allocation failure:", לשחרר את כל הזיכרון שהוקצה עד כה בתכנית, ולצאת מהתוכנית עם EXIT_FAILURE. כרגיל, אין להשתמש ב-exit().
- אם אפשרי, תעדיפו תמיד לעבוד עם int/long מאשר float/double. ניתן לפתור את התרגיל כולו בעזרת שימוש במספרים שלמים בלבד.
- **אין להשתמש ב-**vla, כלומר מערך השמור במחסנית שגודלו נקבע ע״י משתנה. שימוש שכזה יגרור הורדת ציון.

9 נהלי הגשה

- תרגיל זה הינו התרגיל המסכם של שפת C. יש לתרגיל שני חלקים, החלק הראשון מהווה הכנה לחלק השני. קובץ זה מהווה הוראות לחלק השני של התרגיל. אנו ממליצים שלא להתחיל לממש את החלק השני לפני שאתם עוברים את ה-presubmit של החלק הראשון.
- קראו בקפידה את הוראות חלק זה של התרגיל. **זהו תרגיל מורכב ולכן אנו ממליצים להתחיל לעבוד עליו כמה שיותר מוקדם.** זכרו כי התרגיל מוגש ביחידים, ואנו רואים העתקות בחומרה רבה!
- יש להגיש את התרגיל באמצעות ה-git האוניברסיטאי ע"פ הנהלים במודל.
- כחלק מהבדיקה תבדקו על סגנון כתיבה.
- מכיוון שהתרגיל נבדק על מחשבי האוניברסיטה, עליכם לבדוק כי הפתרון שלכם רץ ועובד גם במחשבים אלו.
- כשלון בקומפילציה או ב-presubmit יגרור ציון 0 בתרגיל.
- נזכיר כי חלק זה של התרגיל מהווה 75% מהציון הסופי של התרגיל.
- **מועד הגשה של חלק זה:** יום ה' 25 למאי ב-22:00

בונוס 5+: הגשה עד יום ה׳ 18 למאי ב-22:00

בונוס של 1+/2+/3+ נקודות: הגשה יום/יומיים/שלושה ימים מראש (כרגיל)