**עבודת בית 2 - מעשי**

מתרגלת אחראית: מר גנאדי קוגן

תאריך פרסום: 27.11.2022

תאריך הגשה: 8.12.2022, 23:59

**הנחיות כלליות:**

* קראו היטב את השאלות.
* **ניתן להגיש את העבודה בזוגות או ביחידים.**
* הגשת העבודה תהיה דרך אתר הקורס במודל.
* העבודה תמומש בשפת תכנות C.
* יש להגיש תכנית המכילה שני קבצים (קובץ הגדרות - קובץ עם סיומת .h, קובץ המכיל כל גופי פונקציות עם סיומת с., אין צורך בהגשת main.c).
* יש לכווץ כל קבצי התוכנית לקובץ zip.
* שם הקובץ שיוגש למערכת ההגשה יהיה מורכב מת"ז של המגיש/ים.

לדוגמה:

111111111.zip - עבור הגשה ביחיד

111111111\_22222222.zip - עבור הגשה בזוג

* לפני ההגשה יש להשתייך לקבוצת ההגשה. במקרה של הגשה בזוגות, רק אחד מבני הזוג יגיש את העבודה במודל**.**
* חובה לתעד את הקוד.
* במידה ומצאתם טעויות בעבודה או בקשות להבהרות, אנא שלחו מייל ל- genadko@ac.sce.sc.il
* איחור במועד ההגשה יגרור הורדה של ציון, 5 נק' לכל יום איחור או חלק ממנו.  
  בכל מקרה לא יהיה ניתן להגיש מעבר ל-3 ימי איחור ממועד ההגשה המקורי.  
  במקרים חריגים בלבד יש לפנות למרצה כדי לקבל אישור על הגשה באיחור.
* שאלות לגבי העבודה יש לשאול בפורום באתר הקורס ("מודל") או בשעות קבלה של המתרגל האחראי **בלבד**. ***אין לשלוח שאלות במייל המתרגלות או המרצה.***
* **העבודה תיבדק באופן אוטומטי לפי הפלט אשר התוכניות שלכם תדפיס למסך. לכן, יש להקפיד על ההוראות ולבצע אותן במדויק. כל הדפסה אשר אינה עונה באופן מדויק לדרישות המופיעות בעבודה (כולל שורות, רווחים, סימני פיסוק, או כל תו אחר מיותר, חסר או מופיע בסדר שונה מהנדרש), תגרור הורדה בציון.**

**כללי:**

* עליהם לכתוב פונקציות הבאות המטפלות בעץ בינארי:
  + - ניתן לבנות ולהשתמש בפונקציות עזר.

**שאלה 1 (10 נקודות):**

כתוב/י פונקציה **node \*randomTree()**. פונקציה בונה ומחזירה עץ בינארי אקראי על ידי החזרת מצביע לשורש העץ. מספר הצמתים וערכים בצמתים יוגרלו. כמקרה פרטי, הפונקציה יכולה להחזיר גם עץ ריק. לצורך ההגרלה תשתמשו בפונקציה rand().

**שאלה 2 (10 נקודות):**

כתוב/י פונקציה רקורסיבית **void deleteTree(node \* root)** המקבלת מצביע לשורש של עץ בינארי ומוחקת את העץ (כולל טיפול בשיחרור כל הזיכרון שעץ בינארי תפס).

**שאלה 3 (10 נקודות):**

בהינתן עץ בינארי ומספר שלם חיובי k, כתוב/י פונקציה רקורסיבית   
 **void printNodeAtDistance(node \*root, int k)** שמדפיסה את כל הצמתים במרחק k מהשורש (צמתים בעלי עומק k).

ניתן להניח שפרמטר k תקין ואין צורך לבדוק זאות.

לדוגמה: עבור עץ בינארי A שבאיור ו- k=2, פונקציה מדפיסה את הצמתים הבאים: 1, 4, 8, 11.  
יש להדפיס את הצמתים בסדר הופעתם ברמה, משמאל לימין עם **רווח יחיד ביניהם**.

             5 Binary tree: A

            /  \

           /    \

          3      9

         / \     / \

        1   4   8   11

       / \     /      \

     -3   2   7        12

k=2

output: 1 4 8 11

**שאלה 4 (10 נקודות):**

כתוב/י פונקציה רקורסיבית **void mirrorify(node \*root, node \*\*mirror)** המקבלת מצביע לשורש עץ בינארי המקורי ומצביע לשורש עץ בינארי החדש שיש לבנות. על הפונקציה לבנות עץ חדש שהוא "שיקוף מראה" של העץ המקורי, כלומר, כל הבן הימני הופך להיות בן השמאלי ולהפך. ראו דוגמה עבור עץ בינארי B בעמוד הבא.

Input:

5 Binary tree: B

/ \

3 6

/ \

2 4

Output:

Inorder of original tree: 2 3 4 5 6

Inorder of mirror tree: 6 5 4 3 2

Mirror tree will be:

5

/ \

6 3

/ \

4 2

**שאלה 5: (2 נקודות):**

כדי לבדוק את הנכונות הפונקציה משאלה 4 כתוב/י פונקציה רקורסיבית

**void inOrderTraversal(node \*root)** המקבלת מצביע לשורש עץ בינארי ומבצעת את הסריקה inorder.

כתוב/י עוד שתי פונקציות נוספות:

**void preOrderTraversal(node \*root)** – פונקציה רקורסיבית המקבלת מצביע לשורש עץ בינארי ומתבצעת את הסריקה preorder של העץ.

**void postOrderTraversal(node \*root)** – פונקציה רקורסיבית המקבלת מצביע לשורש עץ בינארי ומתבצעת את הסריקה postorder של העץ.

**שאלה 6 (10 נקודות):**

כתוב/י פונקציה רקורסיבית **node \*insert(node \*root, int val)** המקבלת מצביע לשורש של עץ בינארי וערך val. הפונקציה מוסיפה צומת חדש עם הערך החדש לפי אלגוריתם הבא:

1. Create a new node and assign values to it.

2. Insert(node, key)

     i) If root == NULL,

         assign the new node to be a root of the tree

     ii) if root=>data < key

         call the insert function with root=>right and assign the function’s return value in root=>right.

        root->right = insert(root=>right, key)

     iii) if root=>data > key

         call the insert function with root->left and assign the function’s return value in root=>left.

         root=>left = insert(root=>left, key)

3. Finally, return the pointer to the current root

**שאלה 7 (12 נקודות):**

כתוב/י פונקציה רקורסיבית **bool** **isFull(node\* root)** הבודקת האם עץ הוא עץ מלא (Full Binary Tree). אם העץ מלא, הפונקציה תחזיר true, אחרת היא תחזיר false. הערה: עץ ריק הוא גם עץ מלא. לדוגמה, נעצים הבאים הם עצים מלאים.

A picture containing graphical user interface

Description automatically generated

**שאלה 8 (12 נקודות):**

כתוב/י פונקציה רקורסיבית **void printLeaves(node\* root)** המקבלת מצביע לשורש עץ בינארי ומדפיסה את כל עלים בעץ. **יש להדפיס את העלים לפי סדר הופעתם משמאל לימין עם רווח יחיד ביניהם.** למשל, עבור עץ שבציור הפונקציה תדפיס (משמאל לימין): 42 30 40 9.



**שאלה 9 (12 נקודות):**

כתוב/י פונקציה **רקורסיבית** **int depth(node \*root)** המקבלת מצביע לשורש עץ בינארי ומחשבת את עומקו של עץ ־ אורך המקסימלי במסלול מהשורש עד לעלה.

**שאלה 10 (12 נקודות):**

כתובי פונקציה **int levelStatistics(node\* root, int level)** המדפיסה מפתחות של כל הצמתים ששייכים לרמה level **וגם** מחשבת ומחזירה את מספר הצמתים ברמה.

לדוגמה עבור העץ שמשאל:

* + לרמה 0 – הפונקציה מדפיסה 20 ומספר צמתים 1
  + לרמה 2 – הפונקציה מדפיסה מפתחות (משמאל לימין):

30 44 17 10 ומספר צמתים 4

**יש להדפיס את המפתחות משמאל לימין עם רווח יחיד ביניהם**

**שימו לב:** יש לבדוק תקינות פרמטרים שהפונקציה מקבלת, כלומר אם level אינו חוקי, יש להדפיס הודעת שגיאה “Not a valid level”.

**עבודה נעימה!**