שפות תכנות - ש.ב. 3

עמרי גיא 065982415

גיא שקד 036567055

27 בדצמבר 2010

1. (א) כל איבר מסוג a superList' יכול להיות או מסוג a' או מסוג a superList list'. כך מתאפשר קינון של רשימות מכל עומק.

   (ב) השתמשנו בפונקציית עזר אשר מקבלת סופר רשימה ומספר רמה ולכל אטום ברשימה אותה קיבלה מחזירה צמד של האיבר ומספר הרמה. במידה וקיימת רשימת על נוספת מקוננת, תיקרא פונקציית העזר בשנית עם מספר רמה גדול באחד.

2. (א) הטיפוס abtree מכיל איבר ראש ושני בנים. אם איבר הראש הינו מסוג a' אזי גם שני הבנים יכולים להיות בעלי איבר ראש שכזה ולכן גם הבנים יהיו מאותו הטיפוס. אך במידה ואיבר הראש הינו מסוג b' אזי גם בניו מאולצים להיות מסוג b' בלבד על ידי הגדרתם להיות מסוג abtree ('b,'b). בנוסף, עלי העץ בכל מקרה הינם האיבר unit.

   (ב) בדומה מאוד לסעיף הקודם מוגדר גם abctree רק שהפעם האילוץ לגבי הבנים הינו הדרגתי - קיימים שלושה סוגי עצים: אלו המכילים את שלושת הטיפוסים, אלו המכילים את b' i-c' ואלו המכילים רק את c'.

3. (א) הפונקציה מקבלת שני פסוקים a,b ומחזירה את הפסוק $\neg a|b$ השקול לוגית לפסוק $a \to b$.

   (ב) הפונקציה מקבלת שני פסוקים a,b ומחזירה את הפסוק $(a \wedge \neg b) \vee (\neg a \wedge b)$ השקול ל a XOR b.

   (ג) הפונקציה מקבלת פסוק a ומדפיסה אותו לקונסולה בצורת string.

   (ד) הפונקציה מקבלת פסוק a ומחזירה את צורת ה-NNF שלו לפי הכללים המוגדרים בשאלה.

   (ה) הפונקציה מקבלת שני פסוקים בצורת CNF ומחזירה פסוק CNF השקול לביטוי $a \vee b$. צעדי השכתוב הינם כמוגדר בשאלה.

   (ו) הפונקציה מקבלת פסוק בצורת NNF ומחזירה פסוק שקול לו בצורת CNF.

   (ז) הפונקציה מקבלת פסוק ובודקת האם הוא טאוטולוגיה ע"י האלגוריתם הבא:

      i. העברת הפסוק לצורת NNF.

      ii. העברת הפסוק לצורת CNF.

      iii. חלוקת הפסוק לפסוקים המכילים רק Or (החלקים בין אופרטורי ה-AND בפסוק CNF מכילים רק אופרטורי Or)

      iv. בדיקה האם כל החלקים הנ"ל מכילים כל אחד איזה שהוא אטום ושלילתו וזאת ע"י יצירה של כל הזוגות האפשריים של הליטרלים המופיעים בפסוקית. במידה וכן - נחזיר "אמת". במידה וקיים אפילו חלק אחד כזה שאינו טאותולוגיה - נחזיר "שקר".

4. אין דמיון רב בין go-routines ו-co-routines.

באופן כללי, co-routine היא רוטינה המאפשרת מספר נקודות כניסה בהתאם לנקודות בהן הרוטינה עוצרת את ריצתה. זאת בניגוד לרוטינה רגילה בה יש נקודת כניסה אחת קבועה. ב-co-routine יש בנוסף למנגנון ה-return הקיים ברוטינות רגילות גם מנגנון yield, המחזיר ערך (אופציונלית) לפונקציה הקוראת וממשיך את הריצה שלה, אך גורם לכך שבפע הבאה שה-co-routine תקרא ריצתה תמשיך מהפקודה הבאה אחרי פקודת ה-yield (ולא מתחילתה, כמו ברוטינה רגילה). מנגנון זה מאפשר יצירת איטרטורים, גנרטורים, רשימות אינסופיות, צינורות וכו'...

בשפת Go, go-routine הן פונקציות הנקראות עם המילה השמורה go לפניהן. go-routine רצה במקביל לקוד שקרא לה וערך ההחזרה שלה לא מושם למשתנה. מנגנון ה-go-routine הוא במידה רבה מנגנון של חוטים רזים, ה-go-routine רצה באותו מרחב כתובות כמו הקוד שקרא לה והתקורה שביצירתה והפעלתה אינה גדולה, היא מאפשרת ביצוע משימות במקביל העברת ערכים באמצעות ערוצים (channels) וסנכרון באמצעי סנכרון מקובלים אחרים (מנעולים, למשל).

5. (א) מתכנני השפה בחרו במנגנון simultaneous assignments המאפשר לחלק הימני של ההשמה להיות tuple או רשימת איברים, משערך את כולו ואז משים את האיבר ה־n לאיבר ה־n ברשימת האיברים שמופיעה בצד השמאלי. הדרישה היא שמספר האיברים בשני הצדדים (או ־ מספר האיברים ב־tuple שבצד ימין) יהיה זהה, והטיפוסים ה־nים בשני הצדדים יתאימו (לכל n). מנגנון זה מתיישב היטב עם תכונות ה־strong typing שבשפה, מממש את היתרונות שבהשמה סימולטנית (למשל ־ החלפה בין משתנים ללא צורך במשתנה עזר) והשימוש במזהה הריק (_) מאפשר להשתמש רק בחלק מהמשתנים שברשימה מצד ימין מבלי לפגוע בדרישות התאמת מספר המשתנים והסדר בינהם.

(ב) ניתן למשל לחשב סדרת פיבונאצ'י. בדוגמא הבאה ־ קוד המחשב את מספרי פיבונאצ'י עד ל־1000 ־

```
first,sec := 0,1;
for (sec < 1000) {
  fmt.Println(first);
  first, sec = sec, first+sec;
}
```