**ממ״ן 14**

**שאלה 1:**

1. סכימת התרגום הנדרשת:
2. סכימת תרגום כנדרש (למען הפשטות: ערך ברירת מחדל של כל התכונות הינו 0) משתמשת בתכונות הבאות:

|  |  |
| --- | --- |
| כמה תווים בערך המשתנה A/B צריך להימנע מלהדפיס. | A.toSkip/B.toSkip |
| כמה תווי ‘a’ ישנם בערך המשתנה A/B. | A.length/B.length |
| מונה לכל תו a,b,c בהתאמה – כמה הופעות של התו מופיעות במחרוזת שנקראה משמאל. | S.aBefore/S.bBefore/S.cBefore |
| מונה לכל תו a,b,c בהתאמה – כמה הופעות של התו מופיעות במחרוזת שנקראה כולל המשתנה עצמו (עדכון מונים) | S.aNext/S.bNext/S.cNext |

**שאלה 2:**

הערות: tail, cons, greater, sum, max כולן עושות על ערך רשימה את מה שהפעולה עצמה עושה כמוגדר במטלה. Int(NUMBER) – הכוונה היא להפיכת הייצוג של מספר מייצוג כמחרוזת לערכו הנומרי. Copy/append מעתיקה ומוסיפה איבר לרשימה קיימת, בהתאמה.

**שאלה 4:**

1. הערה: אני מניח שכל המשתנים בתוכנית מוגדרים כint-ים (כלומר, זה רק חלק מתוכנית שמתחילה לפני כן. הוספתי halt כדי שזרימת התוכנית תהיה ברורה וודאית).
2. RLSS not\_geq j 5 // if j < 5 is like not (j >=5)
3. JMPZ not\_geq 5 // go to while if j >= 5
4. IASN z 17 // z = 17
5. JMP 9 // halt
6. RLSS continueloop z 30 // while (z<30)
7. JMPZ continueloop 9 // break if condition doesn’t apply
8. IADD z z b // z += b
9. JMP 5 // possibly another iteration
10. HALT
11. על אותו משקל (מובן שניתן לעשות אופטימיזציה לקוד, אבל לא התבקשתי כאן)  
    אופציה ראשונה:
12. IMLT tmp z 10 // tmp:=z\*10
13. IEQL casejmp tmp 10 // check z\*10==10
14. JMPZ casejmp 6 // next case (20)
15. IASN c 100 // c = 100
16. JMP 8 // fall-through
17. IEQL casejmp tmp 20 // check z\*10==20
18. JMPZ casejmp 10 // next case (default)
19. IASN c 200 // c = 200
20. JMP 11 // break
21. IASN c 300
22. HALT

אופציה שנייה:

1. IMLT tmp z 10 // tmp:=z\*10
2. INQL casejmp tmp 10 // check z\*10!=10
3. JMPZ 7 // execute case 10
4. INQL casejmp tmp 20 // check z\*10!=20
5. JMPZ 8 // execute case 20
6. JMP 10 // execute default
7. IASN c 100 // case: 10
8. IASN c 200 // case: 20
9. JMP 11 // break
10. IASN c 300 // case: default
11. HALT