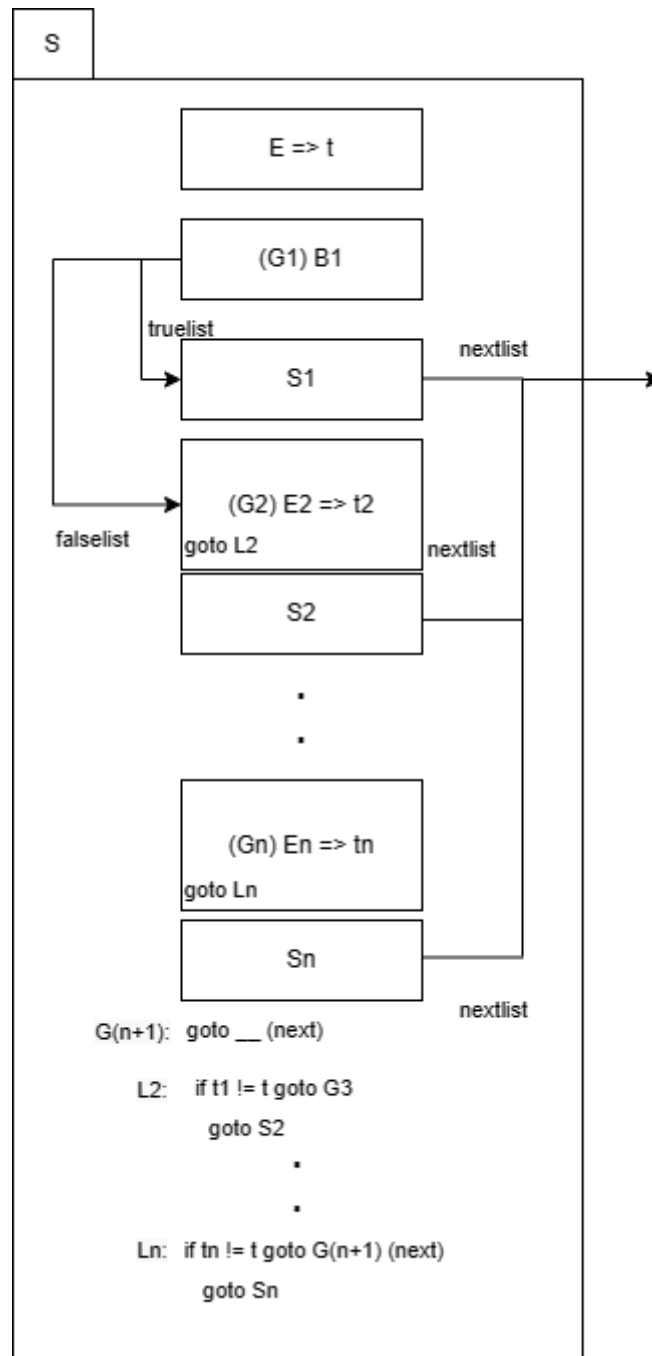


## תרגיל 4

### שאלה 1

.א.



ב. הסבר תכונות:

S: nextlist .  
E: place .  
B: truelist, falselist.  
C: after\_g , after\_s, exp\_address, nextlist, type, value.  
CL: after\_g\_list (stack), after\_s\_list (stack) exp\_address\_list (stack),  
value\_list(stack), nextlist .  
G: truelist, falselist, type, value, next.

S, E, B – כמו בהרצאה.

G:

- א. Truelist: במקרה של E – ריק.  
במקרה של B – רשימת כתובות הקפיצה אותה צריך למלא ב S שלאחר התנאי.  
ב. Falselist: במקרה של E – ריק.  
במקרה של B - רשימת כתובות הקפיצה אותה צריך למלא ב G הבא במידה והתנאי נכשל.  
ג. Type – סוג ה Guard.  
ד. Value – שם המשתנה במקרה וזה exp.  
ה. Next – כתובת פקודת הקפיצה שמופיעה במקרה של E, שאמורה להתמלא בכתובת בה יש השוואת expressions.

C:

- א. After\_g – שומר את הכתובת אחרי ה Guard.  
ב. After\_s – שומר את הכתובת אחרי ה S (בדיוק לפני ה guard הבא).  
ג. Exp\_address – שומר את רשימת הקפיצות שצריך למלא כדי לקפוץ להשוואת exp (אם מדובר ב guard מסוג B אז יהיה ריק)  
ד. Nextlist – הכתובת שאליה ממשיכים לאחר ביצוע S.  
ה. Type – סוג ה guard.  
ו. Value – שם המשתנה של ה E (אם מדובר ב guard מסוג B זה יהיה ערך זבל)

CL:

- א. After\_g\_list – רשימת הכתובות של אחרי ה Guard בכל case.  
ב. After\_s\_list – רשימת הכתובות של אחרי ה S בכל case.  
ג. Exp\_address\_list – רשימת החורים שצריך למלא כאשר ניצור את תנאי הקפיצה.  
ד. Value\_list – רשימת שמות המשתנים של ה expressions.  
ה. Nextlist – הכתובת שאליה ממשיכים לאחר ביצוע ה הנבחר.

## מימוש:

```
G -> is E
{
    G.type = "Exp";
    G.value = E.place;
    G.next = nextquad();
    emit("goto __");
}
```

```
G -> B
{
    G.type = "Bool";
    G.truelist = B.truelist;
    G.falselist = B.falselist;
}
```

```
C -> case G : M S; break ;
{
    C.after_g = M.quad;
    C.after_s = nextquad();
    C.type = G.type;
    C.nextlist = S.nextlist;
    if (G.type = "Exp"){
        C.value = G.value;
        C.exp_address = G.next;
    }
    else if (G.type = "Bool"){
        backpatch(G.truelist, C.after_g);
        backpatch(G.falselist, C.after_s);
    }
}
```

```
CL -> C
{
    CL.after_g_list = newstack();
    CL.after_s_list = newstack();
    CL.nextlist = C.nextlist;
    CL.exp_address_list = newstack();
    CL.value_list = newstack();
    if( C.type = "Exp"){
        CL.exp_address_list.push(C.exp_address);
        CL.value_list.push(C.value);
        CL.after_g_list.push(C.after_g);
        CL.after_s_list.push(C.after_s);
    }
}
```

```
CL -> C CL1
{
    CL.after_g_list = CL1.after_g_list;
    CL.after_s_list = CL1.after_s_list;
    CL.nextlist = merge(CL1.nextlist, C.nextlist);
    CL.exp_address_list = CL1.exp_address_list;
    CL.value_list = CL1.value_list;
    if( C.type = "Exp"){
        CL.exp_address_list.push(C.exp_address);
        CL.value_list.push(C.value);
        CL.after_g_list.push(C.after_g);
        CL.after_s_list.push(C.after_s);
    }
}
```

```

S -> switch (E) CL
{
    S.nextlist = merge ( CL.nextlist, makelist(nextquad()));
    emit ("goto __");
    while(!CL.exp_address_list.empty()){
        after_g = CL.after_g_list.pop();
        after_s = CL.after_s_list.pop();
        exp_address = CL.exp_address_list.pop();
        value = CL.value_list.pop();
        backpatch(exp_address, nextquad());
        emit("if" || value || "!=" || E.place || "goto" || after_s);
        emit("goto" || after_g);
    }
}

```

ג. guard החדש אותו נציע הוא  $G \rightarrow is\ not\ E$ .

נשים לב שעבור הדקדוק הנוכחי באמת אי אפשר לגזור את הguard הזה. כי  $E$  אינו יכול לגרום לביצוע  $S$  בעת אי שוויון  $B$  אינו משווה לערך הביטוי של הswitch כלל.

השינוי הדרוש הוא בגזירה של  $S$ , להוסיף תנאי על השורה:

```
emit("if" || value || "!=" || E.place || "goto" || after_s);
```

כך שאם הטיפוס הוא  $Exp$  השורה תודפס, אם הטיפוס הוא  $nExp$  תודפס:

```
emit("if" || value || "=" || E.place || "goto" || after_s);
```

וגם בבדיקות של האם  $type$  הוא  $exp$  נוסיף בכל המקרים " $or\ nExp$ ".

## שאלה 2:

א. 1. נגדיר את הסריג באופן הבא:

- האיברים: קבוצות של מחרוזות שמייצגות את הביטוי. איבר בדומיין לדוגמה:  $\{ "t1+8", "t1+t2", \dots \}$
- פעולת יחס הסדר  $\sqsubseteq$ : הפעולה תהיה הכלה בכיוון ההפוך  $\supseteq$ .
- פעולת הjoin: חיתוך קבוצות.

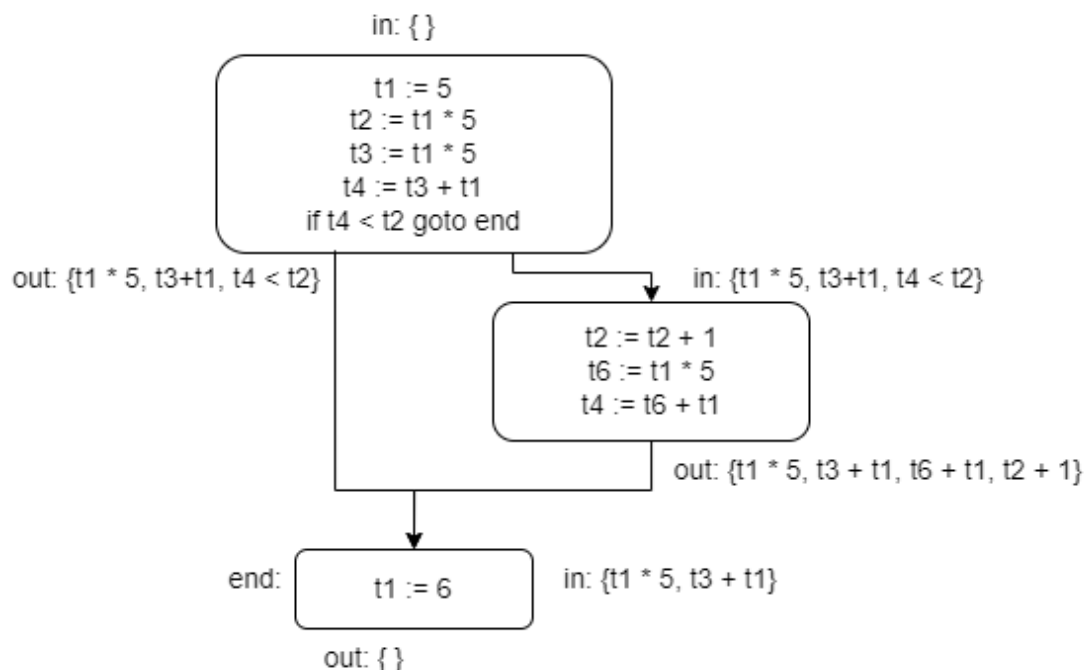
יחס הסדר ופעולת הjoin נבחרו כך מכיוון שנרצה לדרוש שביטוי יהיה זמין מכל מסלול (אחרת יכול להיות מצב שנצטרך לחשב אותו).

2.

Statement	kill	Gen
$X := expr$	$\{ e \in AExpr \mid x \in FV(e) \}$	$\{ e \in AExpr(expr) \mid x \notin FV(e) \}$
goto label	$\phi$	$\phi$
if cond goto label	$\phi$	$AExpr(cond)$
label	$\phi$	$\phi$

הin של המצב הראשון יהיה קבוצה ריקה.

ב.



ג. לכל exp נוסף רשימה של משתנים שהוא כרגע שמור בהם.

השינויים שנבצע בפעולות:

כאשר יש השמה למשתנה, נוריד אותו מהרשימה שהוא מופיע בה (אם הרשימה

התרוקנה נמחק את הביטוי לגמרי) ונוסיף אותו לרשימה של הביטוי החדש (אם לא קיימת כזאת, ניצור רשימה חדשה).

חיתוך: נבצע חיתוך של הרשימות לפי ביטוי ונסיר ביטויים עם רשימות ריקות.

חיפוש: לכל משתנה בביטוי נעבור על כל האופציות להחלפה עם משתנים שנמצאים איתו ברשימה ונבדוק האם ההחלפה נותנת ביטוי קיים. אם כן, מצאנו ☺.