1. הוכח, ללא שימוש ברדוקציה, שהפונקציה MinProg N→N אינה חשיבה.
$MinProg(n) = \min\{|P| : [P()] \geq n\}$

<u>פתרון:</u> נניח שקיימת תוכנית mp(n) המחשבת את MinProg. ונגדיר תוכנית חדשה c().
נסמן $|mp| = N$.

```
function c()
{
    X = 2;
    while (mp(x) <= 9*N)
    {
        X = X*X;
    }

    return X;
}
```

$52 + \log N + N$

תחילה, נשים לב כי $|c| = \underbrace{47}_{\text{אורך הת"כ c}} + \underbrace{\log N}_{\text{אורך המספר N}} + \underbrace{N}_{\text{אורך mp}}$, בהנחה ש-$N>25$ נקבל ש-$|c| \leq 3N < 9N$;

עתה, ע"פ האורך של c והגדרת mp: $MinProg([c()]) < 9N$
אך ע"פ הגדרת c $minProg([c()]) > 9N$ וזו <u>סתירה!</u>

הראנו שבהנחה וקיימת תוכנית המחשבת את minProg אנו מקבלים סתירה ולכן נסיק כי תוכנית זו אינה חשיבה.

<u>מש"ל</u> ☑

הראה רדוקציה והוכחת נכונות.

א. מ- bb ל- halt:

פתרון: נניח שקיימת תוכנית halt המחשבת באופן אמין את הפונקציה halt ונגדיר:

זוג הקדה, לא תוכני
להביא כתוב תוכני
בצורה מפורשת

$$bb(n)=\max\{P() \mid halt(P)=T \wedge |P|\le n\}$$

כלומר, מתוך קבוצת הפונקציות שאורכן קטן או שווה ל-n ועוצרות, תחזיר את הערך המקסימלי המוחזר על ידן.

נכונות: אני מניחה שקיימת פונקציה halt, מספר התוכניות הקטנות/שווה לאורך נתון n הוא מספר סופי. לכן, ניתן לבדוק עבור כל תוכנית האם היא עוצרת (כי halt קיימת). את התוכניות שעוצרות ניתן להריץ ולקבל עבור כל אחת מהן מספר היוצא כפלט. מספר הפלטים המוחזרים גם הוא סופי ולכן ניתן לבחור מהם את המקסימום.

מהרדוקציה אני מסיקה שהפונקציה halt אינה חשיבה היות ואם היא חשיבה הרי גם bb חשיבה (והוכחנו בכיתה שזה לא יכול להיות).

☒

ב. מ- minProg ל- bb.

פתרון: נניח שקיימת תוכנית bb המחשבת באופן אמין את הפונקציה bb ונגדיר:

$$minprog(n)=find(n,0) \text{ where } find(n,k)=\begin{cases} k & bb(k)\ge n \\ find(n,k+1) & o/w \end{cases}$$

נכונות:

① התוכנית עוצרת היות ולכל מספר יש תוכנית באורך מסוים שמחזירה אותו (במקרה הפשוט ביותר תוכנית אשר בה מוקלד ~~מספר~~ return מספר). כמו כן, אני מניחה שקיימת תוכנית bb אמינה ולכן החישוב של bb גם יעצור בכל שלב.

② בכל שלב אני מקבלת את המספר המקסימלי שניתן להפיק מתוכנית באורך k (בהסתמכות על קיום התוכנית bb). היות ובכל שלב אני מבצעת בדיקה האם המספר המקסימלי גדול מ-n (ורק אם לא ממשיכים הלאה ומגדילים את אורך התוכנית ב-1), אכן נקבל בסוף ריצת האלגוריתם את התוכנית באורך המינימלי המפיקה מספר הגדול או שווה ל-n.

☒

מרדוקציה זו נוכל להסיק שהתוכנית bb אינה חשיבה, כיוון שאם הייתה חשיבה התוכנית minProg גם הייתה חשיבה (והוכחנו בכיתה שזה לא).

3. a. בהינתן תוכנית P ללא קלט, האם P מחזירה 2008?

<u>פתרון:</u> הבעיה אינה כריעה. נוכיח ברדוקציה. נניח שקיים פרדיקט כזה הנקרא return2008?(P) ונראה חישוב של $halt_0$ באמצעותו:

$$halt_0(P) := return2008?\left(\underbrace{\lambda. \begin{cases} 2008 & P()=P() \\ 2008 & o/w \end{cases}}_{\text{נסמן כ-}g}\right)$$

<u>הוכחת נכונות:</u>

① במידה והתוכנית P() עוצרת, ערכה בודאי שווה לעצמו, ובכל מקרה התוכנית g תחזיר את הערך 2008. ערך זה יועבר לפרדיקט return2008 שיחזיר T ולכן הערך של $halt_0$ יהיה T גם כן.

② אם התוכנית P() אינה עוצרת, גם g() תכנס ללולאה אינסופית, כלומר לא תחזיר 2008 ולכן return2008? יחזיר F ומכיון שכך $halt_0$ יחזיר F כנדרש.

③ נשים לב שבכל מקרה $halt_0$ לא תכנס ללולאה אינסופית כיון שהנחנו ש return2008? כריעה ולכן לכל קלט היא עוצרת ומחזירה T או F.

הראנו שניתן לחשב את $halt_0$ באמצעות הפרדיקט return2008? אך כפי שלמדנו בשיעור $halt_0$ אינה תוכנית חשיבה. זו <u>סתירה</u> ומכאן נסיק שהבעיה "האם התוכנית P מחזירה 2008?" אינה כריעה.

☒ מש"ל

בהינתן 2 תוכניות P_1 ו-P_2 ללא קלט, האם ל-P_1 ו-P_2 יש אותו פלט? b.

<u>פתרון:</u> הבעיה אינה כריעה. נוכיח ברדוקציה. נניח שקיים פרדיקט הפותר את הבעיה ונקרא לו returnTheSame(P_1, P_2) ונראה חישוב של $halt_0$ באמצעותו:

$$halt_0 := returnTheSame\left(\underbrace{\lambda.[f()=f()]}_{\text{נסמן כ-}g}, \underbrace{\lambda.T}_{\text{נסמן }h}\right)$$

<u>הוכחת נכונות:</u>

① במידה והתוכנית f() עוצרת, ערכה בודאי יהיה שווה לעצמו והתוכנית g תחזיר את הערך T. ערך זה שווה לתוכנית h. לכן, returnTheSame יחזיר T וזה הערך של $halt_0$ וכך כנדרש.

② אם f() אינה עוצרת, g גם לא תחזיר וערכה ($\perp$) יהיה שונה מהערך של h (T). לכן, returnTheSame יחזיר F וזה הערך של $halt_0$ וכך כנדרש.

③ נשים לב שבכל מקרה $halt_0$ לא תכנס ללולאה אינסופית כיון שהנחנו ש returnTheSame כריעה ולכן לכל קלט היא עוצרת ומחזירה T או F.

הראנו שניתן לחשב את $halt_0$ באמצעות returnTheSame אך, כאמור, $halt_0$ אינה תוכנית חשיבה. זו <u>סתירה</u> ומכאן נסיק שהבעיה שלפנינו אינה כריעה.

☒ מש"ל

4.

a. הוכחה שהפונק' MaxSteps אינה חשיבה.

<u>בתרון:</u>

נגדיר $|MaxSteps| = N$

```
function Ms
{
   for i=1  to (MaxSteps(100N) +1)
   {
      Print("Hello");
   }
}
```

נשים לב כי $|Ms| \le 100N$ כי $|Ms| = N + \log(100N+1) + 23 \le 100N$

לכן, $maxSteps(|Ms|) \le maxSteps(100N)$ (כי האורך של Ms קטן מ- 100N).

מצד שני, ע"פ ההגדרה של Ms:

$steps(Ms) > maxSteps(100N)$ (כאשר steps(P) הוא מספר הצעדים שהפונקציה P מבצעת).

קיבלנו סתירה ולכן הפונקציה אינה חשיבה.

☒

שים לב! מוכיח אם"ם אך אין

b. נניח ש maxSteps חשיבה. אם היא חשיבה, ניתן באמצעותה להגדיר את הפונקציה halt. כראה לעיל.

$halt(P, X) := S(P, X, maxSteps(|P|)+1)$

<u>הוכחת נכונות:</u> אני מריצים את תוכנית P עם הערך X באמצעות הסמפר, מספר שלבים הגדול ב-1 ממספר השלבים המקסימלי עבור תוכניות באורך P. אם P אכן עוצרת, היא תהיה חייבת להסתיים גם במהלך הסמפר (כי לא יתכן שהיא תמשיך ותעצור לאחר מספר שלבים הגדול מהמספר המקסימלי). לכן, אם תעצור טבעית, אזי P(x) עוצרת אחרת P(x) נכנסת ללולאה אינסופית.

נגדיר כעת תוכנית חדשה:

```
findN(a,b,c) :=   n=2;
                  while (a^n+b^n ≠ c^n) { n++;}
                  return n;
```

תוכנית זו רצה עד שהיא מוצאת n מתאים ובמקרה זה עוצרת.

כמו כן, נגדיר את הפונקציה הבאה:

```
checkABC:=   for (every a,b,c) {
```

תוכנית זו בודקת עבור כל קלט a,b,c האם find עוצרת עבור כולם
לה. אם כן, היא עוצרת את הבדיקות (כיוון שאם find עוצר עבור הקלט
מסויים, הפרכנו למעשה את המשפט של פרמה), אחרת היא ממשיכה לקלט הבא.

כעת נגדיר את תכנית פרמה:

```
fermal():= !halt(checkABC);
```

תוכנית זו מחזירה T אם checkABC אינה עוצרת (כלומר, משפט פרמה נכון)
ו-F אחרת (כי מצאנו דוגמה שסותרת את המשפט).

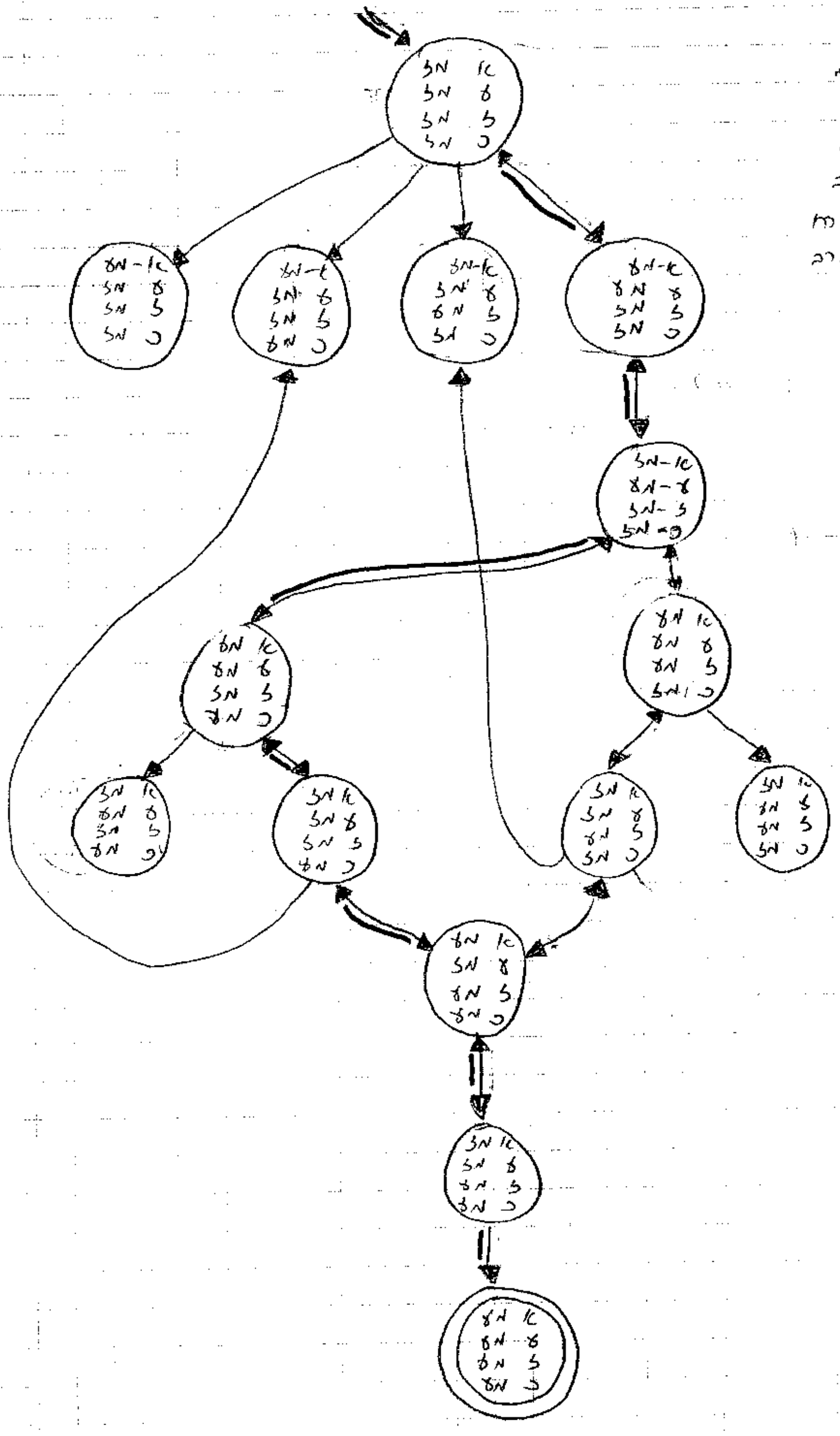
מש״ל

5. רשום STS לבעיה. והראה שקיימת ריצה סופית מקבלת.

<u>פתרון:</u> <u>מצב התחלתי:</u> אדם, עז, זאב וכרוב בצד המזרחי של הנהר.

<u>מצב סיום:</u> אדם, עז, זאב וכרוב בצד המערבי של הנהר.

<u>מקרא:</u>
א - אדם
ע - עז
ז - זאב
כ - כרוב
מז - מזרח
מע - מערב



<u>ריצה סופית מקבלת</u> - קיימות 2 ריצות סופיות מקבלות. הן מסומנות בגרף בצבעים כתום וירוק.

6. הגדר STS למכונת הכפלה פי 2 שהוגדרה בשיעור

<u>פתרון:</u>

$$Q = \{ \langle right, left, ind \rangle \mid right \in \mathbb{N},\ left \in \mathbb{N},\ ind \in \{T, F\} \}$$

כאשר: (1) right מייצג את כמות המחרוזות בתא הימני. $\langle 0,0 \rangle$

(2) left מייצג את כמות המחרוזות בתא השמאלי.

(3) ind הוא אינדיקטור אשר מקבל T לאחר ש-right הוא כבר 0 (אחרת האינדיקטור שווה ל-F).

$$I = \{ \langle right, 0, F \rangle \mid right \in \mathbb{N} \}$$

מצבי ההתחלה הם כל המצבים בהם האינדיקטור הוא F והתא השמאלי שווה ל-0.

$$F = \{ \langle right, 0, T \rangle \mid right \in \mathbb{N}_{even} \}$$

מצבי הסיום הם כל המצבים בהם האינדיקטור הוא T והתא השמאלי הוא 0.

$$\delta = \{ \langle \langle r1, l1, ind1 \rangle, \langle r2, l2, ind2 \rangle \rangle \mid (ind1 = ind2 = F \wedge r2 = r1 - 1 \wedge l2 = l1 + 2) \vee (ind1 = ind2 = T \wedge r2 = r1 + 1 \wedge l2 = l1 - 1) \vee (ind1 = F \wedge ind2 = T \wedge r1 = r2 = 0 \wedge l1 = l2 \in \mathbb{N}_{even}) \}$$

מצבי המעבר האפשריים הם:

(1) כאשר האינדיקטור הוא F (התא הימני עדיין לא הגיע ל-0), נוריד 1 מהתא הימני ונוסיף 2 לתא השמאלי. נשים לב שלא ייתכן מצב של הורדת 1 מ-0 כיוון שבהגדרת המצבים האפשריים, התא הימני חייב להכיל מספר טבעי.

(2) כאשר התא הימני הוא 0 והאינדיקטור הוא F, נעדכן את האינדיקטור להיות T מבלי לשנות את ערך התאים.

(3) כאשר האינדיקטור הוא T, נוסיף 1 לתא הימני ונוריד 1 מהתא השמאלי.

.7

a.

i. בהינתן STS סופי (STS עם מספר סופי של מצבים), האם ניתן לחשב פרדיקט שיקבע אם קיימת ריצה סופית מקבלת?

<u>פתרון:</u> כן.

נסתכל על ה-STS שקיבלנו כתיאור של גרף מכוון שבו Q הינה קבוצת הקודקודים בגרף ו-δ היא קבוצת הקשתות (כלומר $G=(Q,\delta)$).

כדי לבדוק האם קיימת ריצה סופית מקבלת, נריץ BFS החל מכל אחד מהמצבים שנמצאים בקבוצה ההתחלתית I.

לאחר הרצת ה-BFS, נבדוק האם קיים קודקוד v בקבוצת הסיום F שיש לו ערך $d[v]<\infty$ (כלומר יש מסלול המחבר בין הקודקוד ממנו התחלנו את ה-BFS ל-v).

אם קיים קודקוד כזה, נצא מהלולאה ונחזיר T. אם לא קיים קודקוד כזה נמשיך ל-BFS הבא. כשנסיים להריץ BFS על כל אחד מקודקודי ההתחלה, אם לא מצאנו כזה נחזיר F.

<u>הוכחת נכונות</u>

① אם קיים מסלול של ריצה סופית מקבלת, הוא חייב להתחיל בקודקוד מהקבוצה I ולהסתיים בקודקוד מהקבוצה F. לכן, הרצת ה-BFS והבדיקה של קבוצת הסיום לאחר כל BFS, תהיינה חייבות למצוא מסלול שכזה. במידה ולא קיים מסלול, חייב להיות נתק בין כל קודקודי קבוצת הסיום לקודקודי קבוצת ההתחלה ולכן ערך d שלהם יהיה תמיד ∞.

② התוכנית תעצור היות ומדובר על STS עם מספר סופי של מצבים, כמו כן BFS מוגדר היטב ולכן אין חשש ללולאה אינסופית.

מש"ל ☒

ii. בהינתן STS סופי (STS עם מספר סופי של מצבים) ומספר n, האם ניתן לחשב פרדיקט שיקבע אם קיימת ריצה סופית מקבלת באורך n?

<u>פתרון:</u> כן.

גם כאן נסתכל על ה-STS שקיבלנו כתיאור של גרף מכוון. ע"פ [illegible] ההגדרות הנ"ל נבנה עבור כל צומת מצמתי ההתחלה (נבנה את עץ) כל המסלולים האפשריים באורך n. נשים לב שניתן לבנות עץ כזה כיוון שה-STS סופי וקיימת המגבלה של n צעדים. במידה וקיימות לולאות "נפרוש" גם את כל המצבים האפשריים לביצוע הלולאות כך שמספר הצעדים הכולל לא יעלה על n (לבצע את הלולאה פעם אחת, פעמיים וכו').

בסיום פרישת כל עץ, נבדוק האם אחד מצמתי העץ ברמה ה-n שייך לקבוצת המצבים הסופיים. אם כן, נחזיר כן. במידה וסיימנו בדיקת כל העצים, ולא מצאנו מסלול מתאים, נחזיר לא.

הוכחת נכונות: ① היות ובנינו [illegible] את כל המצבים האפשריים [illegible]

(7.a.ii) ② האלגוריתם יעצור שכן מספר המצבים הוא סופי ולכן מספר מצבי הסיום וההתחלה סופיים. כמו כן, כיוון שמספר הצעדים מוגבל ל-n, לא ניכנס ללולאה אינסופית ולכן האלגוריתם יסתיים בזמן סופי.

מש"ל ☒

b. בנה STS M כך שקבוצת הגדלים של הלשונות הסופיות המתקבלות שלו היא $\{2^n \cdot 3^{bb(n)} \mid n\in\mathbb{N}\}$ או הוכח ש STS כזה לא יכול להתקיים.

<u>פתרון:</u> נראה ש STS כזה לא קיים.

נניח בשלילה שקיים STS כזה ונגדיר את root(n) להיות מסלול ריצה סופית מקבלת כך ש $|root(n)| = 2^n \cdot 3^{bb(n)}$. (נשים לב שלכל ערך של $n\in\mathbb{N}$ ניתן להשיג מסלול ובדרך זו <u>ניתן לחשב</u> <u>את הערך של bb(n)</u>, כיוון שהדרגה זו היא הדרגה יחידה ולכן לפי הערך של 2^n ניתן להשיג לכל ריצה מסלול. כלומר שחלוקה של גודלה ב 2^n תחזיר מספר מהצורה 3^x).

כעת נשים לב כי $\log_3\left(\frac{root(n)}{2^n}\right) = \log_3\left(\frac{2^n \cdot 3^{bb(n)}}{2^n}\right) = bb(n)$.

כלומר, הצלחנו לחשב את הערך של bb(n). אך, כפי שראינו בשיעור, bb היא פונקציה שאינה חשיבה ולכן זו <u>סתירה</u>.

הנחנו שקיים STS כנ"ל וקיבלנו סתירה. לכן נסיק שמערכת כזו אינה קיימת.

מש"ל ☒