

S Language Simulator – README

התוכנה מאפשרת למשתמש להריץ קידודים, פרדיקטים ופונקציות שלמדנו בתרגול 6.

המטרה:

שלי: להעמיק את הנסיון שלי בפיתוח, ואת האחיזה שלי בחומר של קורס חישוביות.
של המשתמש: לראות מימוש של הקידודים, הפונקציות, והפרדיקטים שנלמדו בהרצאה.

התוכנה מתחלקת למודלים:

1. מספרים ראשוניים Primes.
2. גורמים ראשוניים Prime_Factors.
3. קידודים coding.
4. קידוד תוכנית Parse_Program.
5. פענוח תוכנית decode_to_program.
6. פרדיקטים ופונקציות נוספות Extended.

1 מודל Primes.

המודל מייבא ספריה חיצונית בשם sympy, שנועדה לנהל מספרים ראשוניים. במידה והיא לא קיימת במחשב, המודל מייצר ישויות מקומיות להחליף את הספריה.

1. רשימה primes: מחזיקה מספרים ראשוניים באינדקסים המתאימים. היא מקבלת את ערכיה מקובץ מצורף בשם saved_primes.txt.
2. פונקציה prime: מקבלת אינדקס. אם האינדקס קיים ברשימה primes, הערך באינדקס יוחזר. אחרת, הרשימה והקובץ יעודכנו להכיל את כל המספרים הראשוניים עד האינדקס הנתון (בשביל לקצר זמן ריצה לגישות הבאות).
3. פונקציה is_prime: מקבלת מספר ומחזירה אם המספר ראשוני או לא.

2 מודל Prime_Factors.

המודל אחראי לתחזוק מילון של מספרי גדל, שמופיע בקובץ spf.txt המצורף. מטרתו לקצר זמני ריצה בקידוד ופענוח של מספרי גדל.

1. מילון saved_factors: מחזיק מספרים טבעיים ואת הפענוח שלהם לגורמים ראשוניים.
2. פונקציה add_and_extend_list: לשימוש פנימי של המודל.
3. פונקציה check saved_factors: מקבלת מספר טבעי ובודקת אם היא מהווה מכפלה כלשהי של מספרי גדל שכבר מופיעים במילון. הפונקציה מחזירה את תוצאת החילוק של הקלט במספרי גדל שמופיעים במילון, יחד עם הפענוח החלקי שלו.
- דוגמא: אם במילון יש את המספר 10 עם הפענוח שלו [1,0,1] ונרצה לקבל את הפענוח של המספר 30, הפונקציה תחזיר [3, [1,0,1]].
4. פונקציה verify_saved_factors: לאימות המילון, לא בשימוש.
5. פונקציה save_result: מקבלת מספר טבעי ורשימה שמהווה פענוח של המספר. היא שומרת את הקלט במילון המקומי ובקובץ spf.txt לשימוש עתידי.

3 מודל coding.

המודל מכיל פונקציות לקידוד ופענוח כפי שלמדנו בהרצאה.

1. פונקציה pair_encode: מקודדת זוג מספרים למספר יחיד.
2. פונקציה pair_decode: מפענחת מספר לזוג מספרים.
3. פונקציה r,l: מחזירה את המספר הימני או השמאלי של מספר מקודד.

4. פונקציה `list_encode` : מחזירה מספר גדל של רשימה, ושולחת את התוצאה למודל `prime_factors` לשמירה.
5. פונקציה `list_decode` : מקבלת מספר ומפענחת אותה לרשימה, תוך שמירת גילויים חדשים של קידודי גדל במהלך הפענוח.
6. פונקציה `ith` : מקבלת מספר גדל ואינדקס, ומחזירה את המספר במקום של האינדקס בפענוח של מספר הגדל. ניתן לשלוח לפונקציה רשימה טרום-קידוד במקום מספר גדל.
7. פונקציה `lt` : מחזירה את אורך פענוח מספר הגדל. ניתן לשלוח לפונקציה רשימה טרום-קידוד במקום מספר גדל.
8. פונקציה `get_a_b_c` : מפענחת מספר טבעי `<a,<b,c>>` ומחזירה את המספרים `a,b,c`.
9. פונקציה `encode_a_b_c` : מקבלת שלושה מספרים טבעיים ומחזירה `<a,<b,c>>`.

4 מודל `Parse_Program`.

המודל אחראי לקודד תכנית בשפת `S` טהורה למספר טבעי.

1. שלושה ביטויים רגולריים מקומפלים `label_finder`, `if_finder`, `var_finder` : לשימוש פנימי של המודל.
2. פונקציה `encode_instruction` : הפונקציה מקבלת מחזורת של קוד בשפת `S`. תוך שימוש בביטויים רגולריים, הפונקציה מחלצת מהמחרוזות ומחזירה את הערכים `a,b,c` כך שקידוד הפקודה תהיה `<a,<b,c>>`.
3. פונקציה `get_label` : לשימוש פנימי של המודל.
4. פונקציה `encode_file` : מקבלת `path` לקובץ שמכיל תוכנית בשפת `S`, מקודדת את התוכנית, מדפיסה את התוכנית ואת הקידוד שלה, ומחזירה את הקידוד שלה.
5. פונקציה `encode_str` : מקבלת מחרוזת של תוכנית, מקודדת אותה, מדפיסה את התוכנית ואת הקידוד שלה, ומחזירה את הקידוד שלה.

5 מודל `decode_to_program`.

המודל אחראי לפענח מספר תוכנית לתוכנית עצמה.

1. פונקציה `decode_instruction` : מקבלת שלישייה של מספרים `a,b,c` ומחזירה את הפקודה המקודדת על ידי `<a,<b,c>>`.
2. פונקציה `decode_from_num` : מקבלת מספר תוכנית ומפענחת אותו. היא מדפיסה את התוכנית, ומחזירה אותה בתור מחרוזת.
3. פונקציה `decode_from_list` : מקבלת רשימה של מספרים שמהווים קידודים לפקודות, ומפענחת את הרשימה לפקודות עצמן. היא מדפיסה את התוכנית, ומחזירה אותה בתור מחרוזת.
4. פונקציה `get_tag` : לשימוש פנימי של המודל.
5. פונקציה `decode` : פונקציית מעטפת לשתי הפונקציות לעיל. מקבלת ישות, ולפי סוג הישות (רשימה/מספר) שולחת אותה לפונקציה המתאימה.

6 מודל `Extended`.

המודל מכיל את כל הפרדיקטים והפונקציות שהכרנו בתרגול 6.

1. פונקציה `label` : מחזירה את מספר התווית שמופיעה באינדקס הנתון בתוכנית הנתונה.
2. פונקציה `var` : מחזירה את מספר המשתנה שמופיע באינדקס הנתון בתוכנית הנתונה.
3. פונקציה `instr` : מחזירה את סוג הפקודה שמופיעה באינדקס הנתון בתוכנית הנתונה.
4. פונקציה `label_` : מחזירה את מספר תווית הקפיצה שמופיעה באינדקס הנתון בתוכנית הנתונה.
5. פונקציה `term` : מקבלת מצב רגעי ומספר תוכנית ומחזירה אם התוכנית הסתיימה.
6. פונקציה `skip` : מקבלת מצב רגעי ומספר תוכנית ומחזירה אם לפקודה הבאה יש השפעה על המשתנים של התוכנית.
7. פונקציה `incr` : מקבלת מצב רגעי ומספר תוכנית ומחזירה אם הפקודה היא הוספה.
8. פונקציה `decr` : מקבלת מצב רגעי ומספר תוכנית ומחזירה אם הפקודה היא הורדה, ואם ערך המשתנה בפקודה גדול 0.
9. פונקציה `branch` : מקבלת מצב רגעי ומספר תוכנית ומחזירה אם הפקודה היא התניה, ואם היא תתבצע בפועל.
10. פונקציה `succ` : מקבלת מצב רגעי ומספר תוכנית ומחזירה את המצב הרגעי הבא אחריו.

11. פונקציה `init` : מקבלת קלטים ומספר תוכנית ומחזירה את המצב הרגעי הראשוני של התוכנית.
12. פונקציה `snap` : מקבלת קלטים, מספר תוכנית, ואינדקס, ומחזירה את המצב הרגעי ה-`i`.
13. פונקציה `stp` : מקבלת קלטים, מספר תוכנית, ואינדקס, ומחזירה אם עד אותו אינדקס של מצב רגעי, התוכנית הסתיימה.
14. פונקציה `print_snap` : מקבלת מצב רגעי מקודד ומדפיסה את הפענוח שלו.
15. פונקציה `output` : מקבלת קלטים, מספר תוכנית, ואינדקס, ומחזירה את הערך של משתנה הפלט במצב הרגעי ה-`i`.
16. פונקציה `assist` : לשימוש פנימי של המודל. מאפשרת לשלוח רשימות של קידודי פקודות במקום לשלוח מספר גדל לכל הפונקציות של המודל.

דוגמאות לשימוש

התוכנית בסימולציות היא התוכנית של הפונקציה $f(x)=x$

[illegible]

