מת"מ: בדיקת מספרים ראשוניים

בתרגיל זה תכתבו תוכנית שמקבלת מספרים ובודקת האם האם ראשוניים או לא. את התרגיל תבדקו על שרת החוג בעזרת הבודק האוטומטי. אם לא הצלחתם עדיין להתקין סביבת עבודה מתאימה אתם יכולים לכתוב את התרגיל דרך סביבת עבודה אינטרנטית לסי דוגמת https://www.onlinegdb.com/. לאחר שתכתבו את הקוד תוכלו להוריד אותו למחשב שלכם ולהעתיק אותו לשרת החוג (cs.telhai.ac.il).
בצעו את השלבים הבאים אחד אחרי השני. לאחר כל שלב הריצו את התוכנית ובידקו שהיא עושה את מה שאתם מצפים – אל תכתבו את הכל בבת אחת!
המנעו מלהשתמש ב-AI לצורך כתיבת הקוד. העזרו במצגת המבוא בשביל למצוא דוגמאות קוד.

חלק א: כתיבת תוכנית לבדיקת מספרים ראשוניים

1. אם עוד לא יצרתם תיקייה על המחשב שלכם עבור הקורס, צרו תיקייה בשם Matam.
2. תחת Matam, צרו תיקייה בשם Lab2.
3. צרו קובץ בשם main.c תחת Lab2.
4. כתבו תוכנית פשוטה שמדפיסה Hello World למסך. ודאו שהיא עובדת.
5. הוסיפו את הקוד הבא לתחילת main:
   - הגדירו משתנה בשם num מטיפוס unsigned int. אתחלו אותו ל-0.
   - הדפיסו את הטקסט הבא למסך (עם רווח אחרי ה-:)

     `Enter a positive number:`
   - קראו את המספר שהמשתשמת תכניס לתוך num. השתמשו ב-scanf. כדי לקרוא מספר שלם לא שלילי ב-scanf (וגם כדי להדפיס אותו ב-printf) משתמשים ב-%u. ודאו ש-scanf קראה את המספר, הדפיסו הודעת שגיאה וצאו מהתוכנית ע"י return אם לא.
     במקרה שהקלט שהוכנס אינו מספר שלם לא-שלילי הדפיסו:

     `Error: input is not a non-negative number`

     וצאו מהתוכנית עם קוד 1 (return 1). ראו דוגמא לשימוש ב-scanf במצגת המבוא לשימוש ב-scanf.
6. ודאו שהקלט הוא מספר בין 1 ל-2000000. במקרה שהקלט שהוכנס אינו בין 1 ל-2000000 הדפיסו:

   `Error: number must be between 1 to 2000000`

   וצאו מהתוכנית עם קוד 2.
7. שנו את פקודת ה-printf שמדפיסה Hello World כך שתדפיס את המספר שהוכנס. הריצו את התוכנית, ודאו שאתם מצליחים להכניס מספר ולהדפיס אותו למסך. ודאו שאם לא מתקבל מספר שלם חיובי התוכנית מדפיסה את הודעת שגיאה ויוצאת.
8. הוסיפו לתחילת הקובץ include ל-stdbool.h.
9. הגדירו, מעל main, פונקציה בשם is_prime שמקבלת משתנה בשם num מטיפוס unsigned int ומחזירה bool.
10. הוסיפו ל-is_prime קוד שיבדוק האם num ראשוני או לא. מספר הוא ראשוני אם הוא מתחלק ללא שארית רק ב-1 ובעצמו. עבור שני מספרים x ו-y אפשר לבדוק האם x מתחלק ב-y ע"י שימוש במודולו: אם x%y=0 אז x מתחלק ב-y ללא שארית. למשל: 10%2=0 אבל 10%3=1. הקוד שתכתבו לא צריך להיות יעיל. הפונקציה תחזיר true אם המספר הוא ראשוני או false אחרת.

11. הוסיפו ל-main קוד שיקרא ל-is_prime עבור המספר שהוכנס וידפיס:
במקרה שהמספר שהוכנס הוא ראשוני:

```
<n> is a prime number
```

החליפו את <n> במספר שהוכנס.
במקרה שהמספר שהוכנס אינו ראשוני:

```
<n> is not a prime number
```

12. בידקו את התוכנית שלכם וודאו שהיא עובדת. נסו להכניס קלטים לא חוקיים, כולל מחרוזות, מספרים שליליים ומספרים לא שלמים. נסו את התוכנית על הקלטים 0 ו-1 (1 אינו ראשוני!) ואז על כמה מספרים ראשוניים וכמה מספרים לא ראשוניים. לתוכנית שלכם אסור לסיים עם שגיאת זמן ריצה!

חלק ב: בדיקת התוכנית עם הבודק האוטומטי

13. התחברו לשרת החוג דרך הטרמינל של VSCode או בכל דרך אחרת שאתם רגילים אליה.
14. בתיקיית הבית שלכם, צרו תיקייה בשם Matam. בתוך התיקייה צרו תיקייה בשם L2.
15. העתיקו את main.c מהמחשב שלכם לשרת החוג, לתיקייה L2. אם אתם מתחברים דרך terminal של VSCode או דרך Terminal של mac אתם יכולים להעתיק את main.c כך מתוך תיקיית התרגיל שלכם:

```
scp -P 12123 main.c cs12345@cs.telhai.ac.il:Matam/L2
```

החליפו את 12345 במספר ת.ז. שלכם.
16. הכנסו לתיקייה L2 דרך שורת הפקודה.
17. הריצו את הפקודה

```
gcc -Wall -Werror -g -std=c99 -o L2 main.c
```

הפקודה תקמפל את התוכנית. אם אין שגיאות התיקייה תכיל קובץ הרצה בשם L2. הריצו אותו ובידקו שהוא עובד כמו שצריך.
18. הריצו את הפקודה הבאה, החליפו את 12345 במספר ת.ז. שלכם:

```
zip 73_L2_12345.zip main.c
```

19. הריצו את הבודק האוטומטי על התוכנית שלכם:

```
hwcheck 73_L2_12345.zip
```

עבור המעבדה הזו יצרנו 10 טסטים. על המסך תראו הודעות כמו זו עבור טסטים שהצליחו:

```
Running Test: Test1.txt
Passed 1 test(s)
```

והודעות כאלה עבור טסטים שנכשלו:

```
Running Test: Test3.txt
Comparing myOutputTest3.txt.txt against /home/itaish/homework/73/L2/outputTest3.txt
ERROR >>> Please inspect the difference file diff.myOutputTest3.txt.txt
Failed 1 out of 1 test(s)
```

20. הבודק האוטומטי ייצור תיקייה בשם **73_L2_12345** (החליפו את 12345 במספר ת.ז. שלכם). עבור טסטים שנכשלו, חפשו בתיקייה החדשה את הקלט שלהם (בקובץ inputTest<i>.txt, <i> הוא מספר הטסט), את הפלט המצופה מהם (outputTest<i>.txt) ואת הפלט של התוכנית שלכם (myOutputTest<i>.txt). תקנו את התוכנית, מחקו את **73_L2_12345.zip** והריצו שוב את הבודק.
21. לאחר שכל הטסטים עוברים, העתיקו את **73_L2_12345.zip** למחשב שלכם והגישו אותו דרך המודל.

בהצלחה!