Activity 2: the health guiding bot!

Research:

In C programming, "ranges" refer to the defined boundaries or intervals within which variables, array indices, loop counters, or other data elements must operate. Unlike higher-level languages like Python that have built-in `range()` objects, C handles ranges through explicit boundary checking, loop constructs, and conditional logic.there mainly if-else if statement is used

Analysis (methodology):Takes input value(s)

- Defines range intervals
- Checks input against these via if-else or switch-case (if exact values)
- Executes code based on matched range
- Handles exceptions or invalid ranges gracefully

This pattern is widely used in grading systems, category-based classification, decision-making aids, and many beginner to intermediate programming problems in C.

- Think of ranges as intervals on a number line where values belong; the program checks which interval the input falls in.
- Use clear and non-overlapping range conditions to avoid logical errors.
- Remember the order of conditions matters; start checking from the highest or lowest range depending on the problem.
- Range programs often combine loops and conditions if multiple inputs or values need to be categorized.
- Practice with common examples like grading systems, eligibility checkers, or temperature converters to build intuition.

Ideation:

The basic idea is to design the bot which can guide the user by getting input from them and also give them suggestions to improve if any

First of all the inputs regarding the daily activities of the user is collected in the form of the numbers or the ratings

Define Health Parameters to Assess

- Choose key habits impacting health (e.g., exercise frequency, junk food intake, yoga practice, sleep duration).
- Each habit becomes a variable in the program (like `exercise`, `junk_food`, etc.).

2. Set Healthy Thresholds or Ranges

- Decide what counts as healthy or unhealthy levels. For example:
    - Exercise: at least 3 times per week
    - Junk food: no more than 2 times per week
    - Yoga: at least 2 times per week
    - Sleep: at least 7 hours per night
- These thresholds represent scientific or commonly agreed health guidelines.

3. Collect User Input

- Prompt the user to enter their values for each habit.
- Use input functions to gather this data so program can analyze it.

4. Check Values Against Thresholds

- For each habit, compare user input to its healthy range.
- If input falls outside the healthy range, mark the lifestyle as "not healthy" for that aspect.
- Store an advice message or feedback related to improving that habit.

5. Provide Feedback

- After all checks, let the user know if they are maintaining a healthy lifestyle overall.
- If not, print detailed suggestions for each habit that does not meet healthy criteria

Algorithms:

Start

1. Initialize variables:
    - `exercise`, `junk_food`, `yoga`, `sleep` to store user input.
    - `healthy` to 1 (assume healthy initially).
    - `feedback` as a list (or conceptually a set of strings) to store improvement messages.
    - `num_feedback` to 0 to track number of feedback messages.
2. Display prompt and get input:
    - Ask user to enter number of exercise sessions per week; store in `exercise`.
    - Ask user to enter junk food intake per week; store in `junk_food`.
    - Ask user to enter yoga sessions per week; store in `yoga`.
    - Ask user to enter sleep hours per night; store in `sleep`.
3. Check exercise threshold:
    - If `exercise` less than 3:
        - Set `healthy` to 0.
        - Add "Increase exercise to at least 3 times/week" to `feedback`.
        - Increment `num_feedback`.
4. Check junk food threshold:
    - If `junk_food` greater than 2:
        - Set `healthy` to 0.
        - Add "Reduce junk food to max 2 times/week" to `feedback`.
        - Increment `num_feedback`.
5. Check yoga threshold:
    - If `yoga` less than 2:
        - Set `healthy` to 0.
        - Add "Practice yoga at least 2 times/week" to `feedback`.
        - Increment `num_feedback`.
6. Check sleep threshold:
    - If `sleep` less than 7:
        - Set `healthy` to 0.
        - Add "Aim for at least 7 hours of sleep/night" to `feedback`.
        - Increment `num_feedback`.
7. Display results:

- If `healthy` equals 1:
    - Print "You're maintaining a healthy lifestyle! Keep it up!"
- Else:
    - Print "Areas to improve: "
    - For each message in `feedback`:
        - Print the message.

Build:

```
#include <stdio.h>


int main() {
    int exercise, junk_food, yoga, sleep;
    int healthy = 1;
    char *feedback[4];
    int num_feedback = 0;

    printf("Enter exercise sessions per week (1-7): ");
    scanf("%d", &exercise);

    printf("Enter junk food intake per week (1-7, 1=never): ");
    scanf("%d", &junk_food);
```

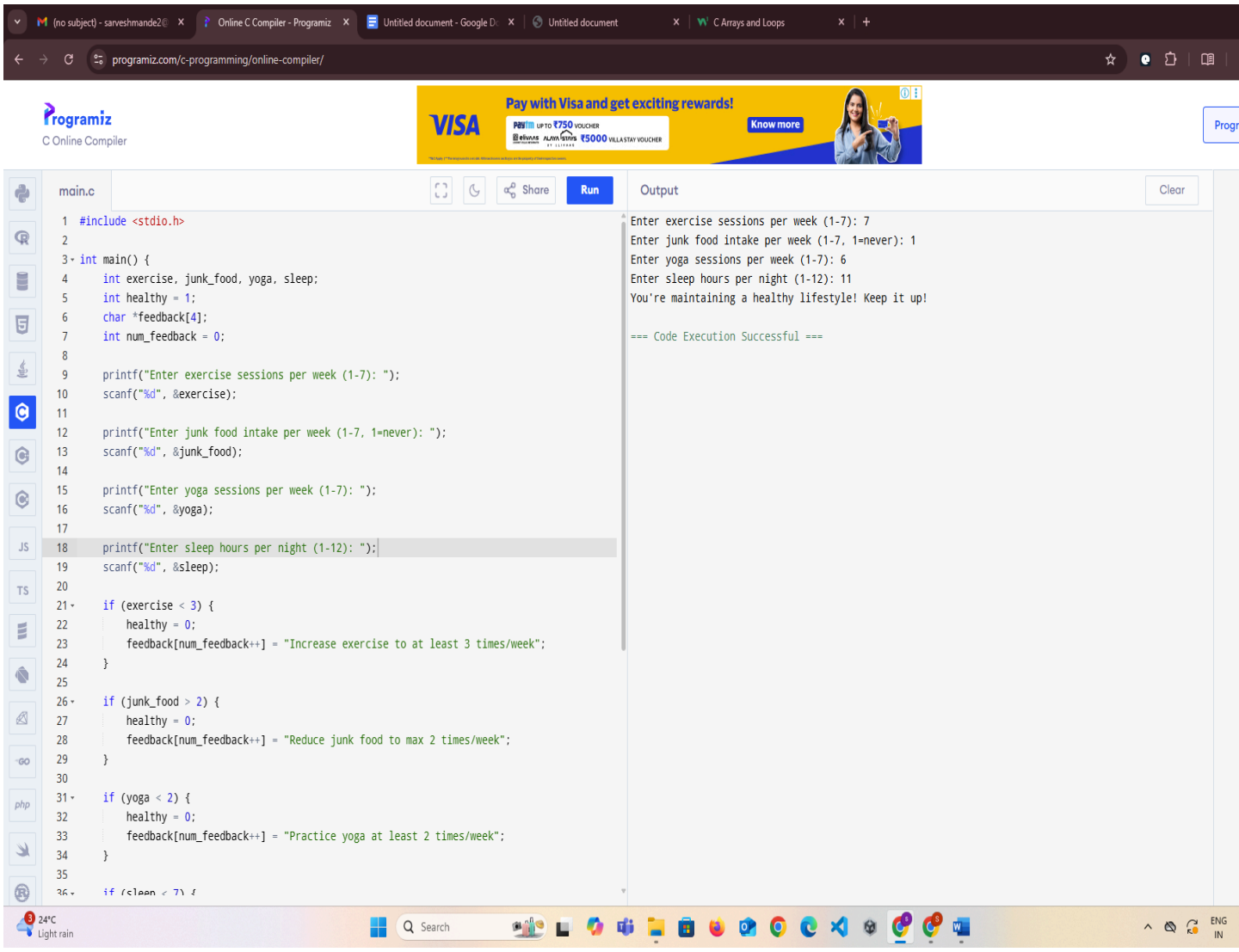
```
    printf("Enter yoga sessions per week (1-7): ");
    scanf("%d", &yoga);

    printf("Enter sleep hours per night (1-12): ");
    scanf("%d", &sleep);

    if (exercise < 3) {
        healthy = 0;
        feedback[num_feedback++] = "Increase exercise to at least 3 times/week";
    }

    if (junk_food > 2) {
        healthy = 0;
        feedback[num_feedback++] = "Reduce junk food to max 2 times/week";
    }

    if (yoga < 2) {
```

```
        healthy = 0;
        feedback[num_feedback++] = "Practice yoga at least 2 times/week";
    }

    if (sleep < 7) {
        healthy = 0;
        feedback[num_feedback++] = "Aim for at least 7 hours of sleep/night";
    }

    if (healthy) {
        printf("You're maintaining a healthy lifestyle! Keep it up!");
    } else {
        printf("Areas to improve:");
      ; for (int i = 0; i < num_feedback; i++) { printf("- %s ", feedback[i]); } }
return 0;
```

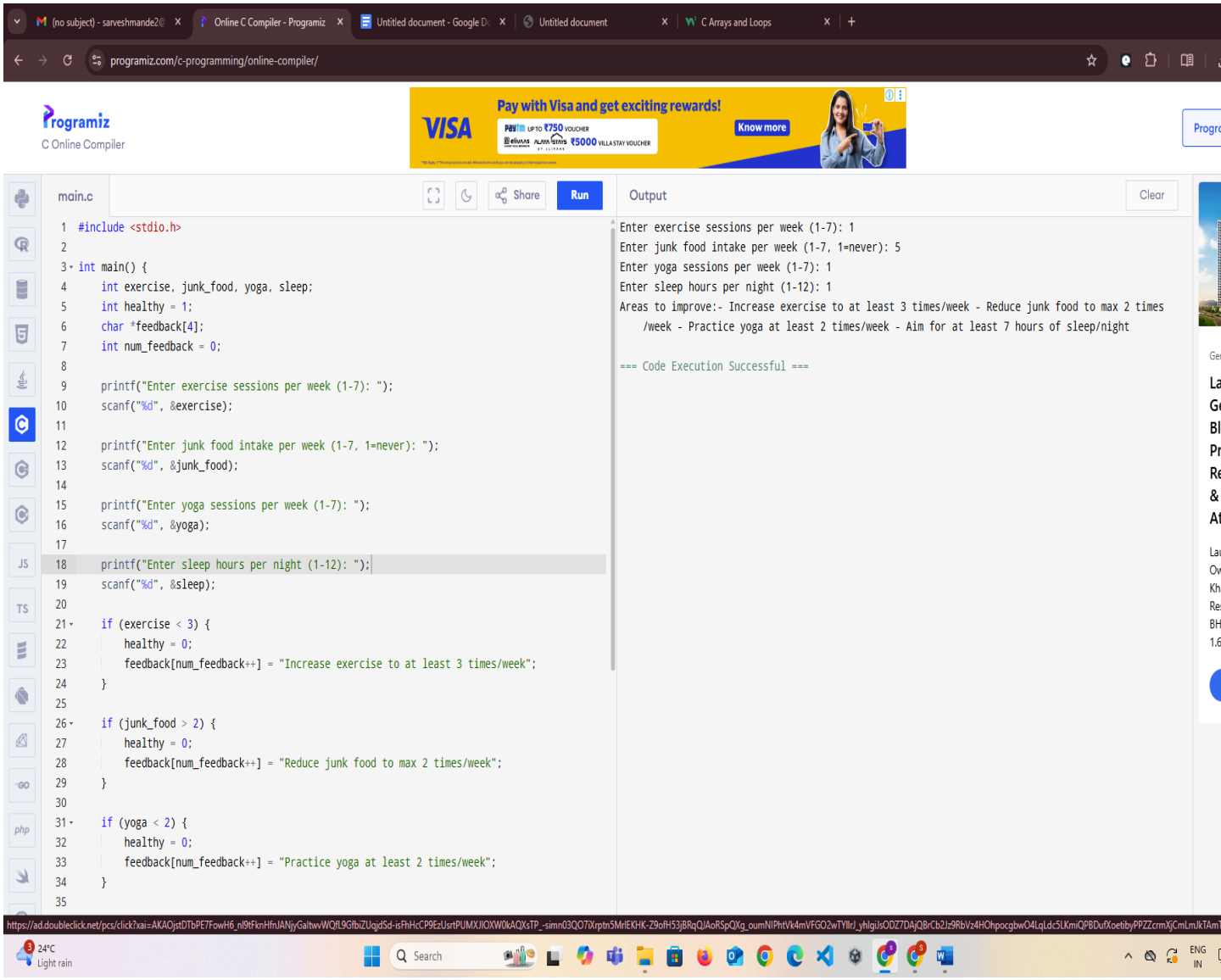
```
}
```

Testings :

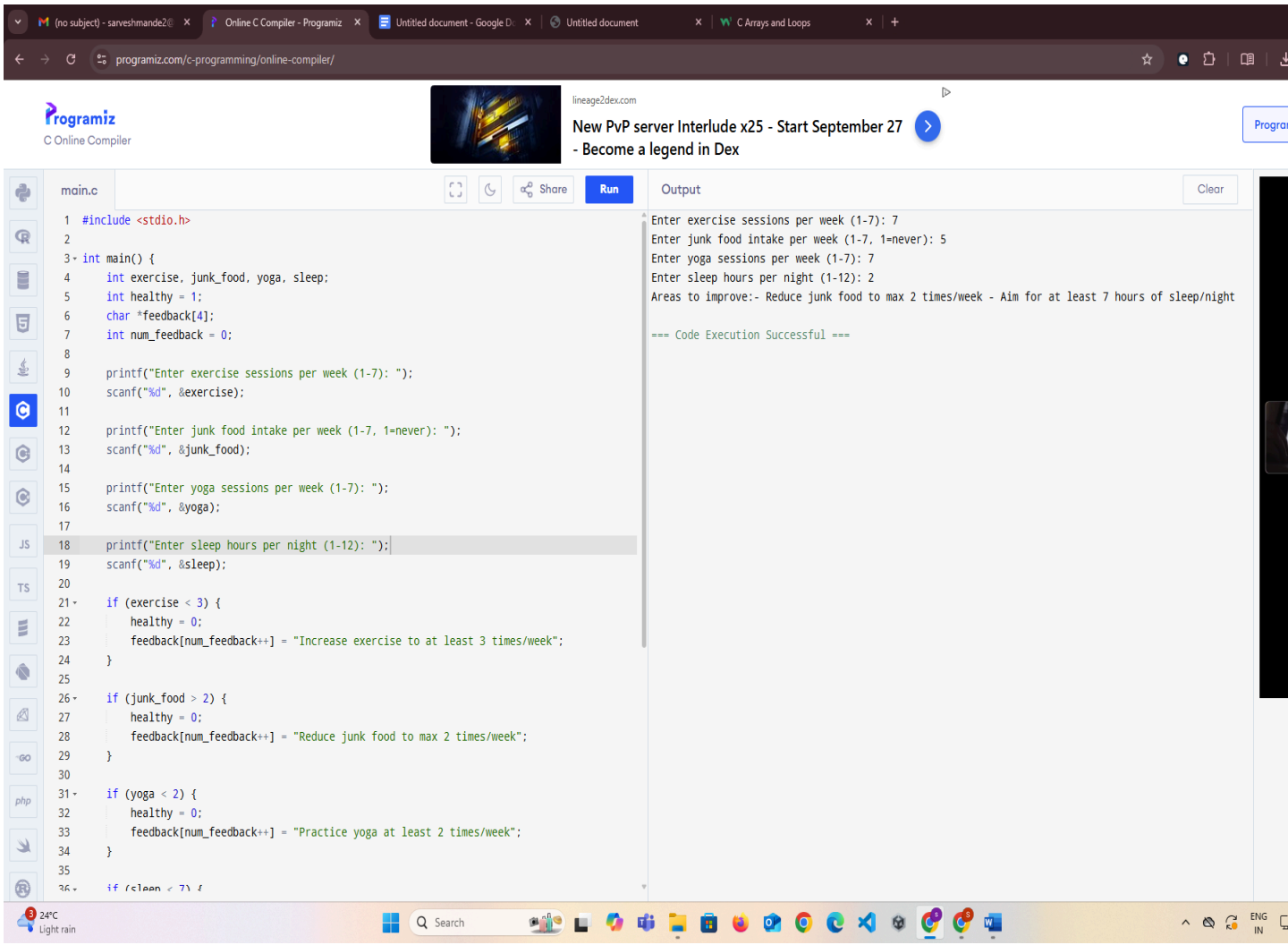
Case 1: perfect health



Case2 totally improper health and areas to improve them



Case3 mix possibility one criteria is good but others are not



https://github.com/sarvesh00719/health-guiding-bot