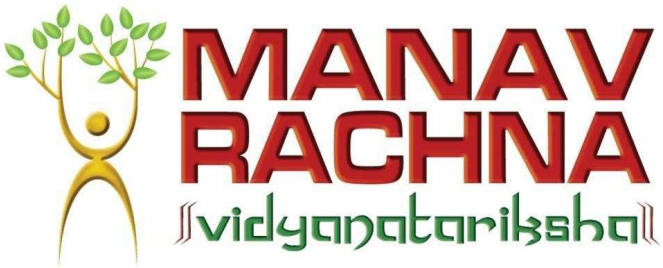
Manav Rachna International Institute of Research and Studies



SCHOOL OF COMPUTER APPLICATION

MCA - B 1st SEM

PROJECT REPORT

(Course Code: 6.0CA101C01H)

SUBMITTED BY:

TINKU JAIN 25/SCA/MCAN/071

ABHISHEK GUPTA 25/SCA/MCAN/072

HIMANSHU JHA 25/SCA/MCAN/ 070

SHIVAM KUMAR 25/SCA/MCAN/ 073

SUBMITTED TO:

Dr. Sachin Sharma

∇ HEALTH TRACKER

CODE: -

```
/* health_tracker.c
 *
 * Personal Health Metrics Tracker and Visualiser
 * Tracks: step count, calories burned, sleep duration.
 *
 * Compile: gcc -std=c11 -Wall -Wextra -O2 health_tracker.c -o health_tracker
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define INITIAL_CAPACITY 16
#define DATE_LEN 11      /* YYYY-MM-DD + null */
#define INPUT_BUF 256

typedef struct {
    char date[DATE_LEN];   /* YYYY-MM-DD */
    int  steps;
    int  calories;
    float sleep_hours;     /* e.g., 7.5 */
} HealthEntry;

static void trim_newline(char *s) {
    if (!s) return;
    size_t len = strlen(s);
    if (len > 0 && s[len - 1] == '\n') s[len - 1] = '\0';
}

static void flush_line_if_needed(const char *buf) {
    size_t len = strlen(buf);
    if (len > 0 && buf[len - 1] != '\n') {
        int ch;
        while ((ch = getchar()) != '\n' && ch != EOF) {
            /* discard */
```

```
        }
    }
}

static int read_line(char *buf, size_t size) {
    if (!fgets(buf, (int)size, stdin)) return 0;
    flush_line_if_needed(buf);
    trim_newline(buf);
    return 1;
}

static int parse_int(const char *s, int *out) {
    char *end;
    long val = strtol(s, &end, 10);
    if (end == s || *end != '\0') return 0;
    if (val < -2147483648L || val > 2147483647L) return 0;
    *out = (int)val;
    return 1;
}

static int parse_float(const char *s, float *out) {
    char *end;
    float val = strtof(s, &end);
    if (end == s || *end != '\0') return 0;
    *out = val;
    return 1;
}

/* Simple YYYY-MM-DD format check (no real calendar validation). */
static int is_valid_date(const char *s) {
    if (strlen(s) != 10) return 0;
    for (int i = 0; i < 10; ++i) {
        if (i == 4 || i == 7) {
            if (s[i] != '-') return 0;
        } else {
            if (!isdigit((unsigned char)s[i])) return 0;
        }
    }
    return 1;
}

static int append_entry(HealthEntry **arr, size_t *count, size_t *cap, HealthEntry entry) {
    if (*count >= *cap) {
        size_t newcap = (*cap == 0) ? INITIAL_CAPACITY : (*cap * 2);
```

```
        HealthEntry *tmp = realloc(*arr, newcap * sizeof(HealthEntry));
        if (!tmp) return 0;
        *arr = tmp;
        *cap = newcap;
    }
    (*arr)[*count] = entry;
    (*count)++;
    return 1;
}

static void add_entry(HealthEntry **arr, size_t *count, size_t *cap) {
    char buf[INPUT_BUF];
    HealthEntry e;

    printf("Enter date (YYYY-MM-DD): ");
    if (!read_line(buf, sizeof(buf))) {
        printf("Input error.\n");
        return;
    }
    if (!is_valid_date(buf)) {
        printf("Invalid date format. Use YYYY-MM-DD.\n");
        return;
    }
    strncpy(e.date, buf, DATE_LEN - 1);
    e.date[DATE_LEN - 1] = '\0';

    printf("Enter step count: ");
    if (!read_line(buf, sizeof(buf))) {
        printf("Input error.\n");
        return;
    }
    if (!parse_int(buf, &e.steps) || e.steps < 0) {
        printf("Invalid step count. Must be a non-negative integer.\n");
        return;
    }

    printf("Enter calories burned: ");
    if (!read_line(buf, sizeof(buf))) {
        printf("Input error.\n");
        return;
    }
    if (!parse_int(buf, &e.calories) || e.calories < 0) {
        printf("Invalid calories. Must be a non-negative integer.\n");
        return;
```

```
    }

    printf("Enter sleep duration (hours, e.g., 7.5): ");
    if (!read_line(buf, sizeof(buf))) {
        printf("Input error.\n");
        return;
    }
    if (!parse_float(buf, &e.sleep_hours) || e.sleep_hours < 0.0f) {
        printf("Invalid sleep value. Must be a non-negative number.\n");
        return;
    }

    if (!append_entry(arr, count, cap, e)) {
        printf("Memory error while saving entry.\n");
        return;
    }

    printf("Entry added for %s.\n", e.date);
}

static void list_entries(const HealthEntry *arr, size_t count) {
    if (count == 0) {
        printf("No entries recorded yet.\n");
        return;
    }
    printf("\n%-4s | %-10s | %-10s | %-10s | %-12s\n",
           "No.", "Date", "Steps", "Calories", "Sleep (h)");
    printf("-----+------------+------------+------------+--------------\n");
    for (size_t i = 0; i < count; ++i) {
        printf("%-4zu | %-10s | %-10d | %-10d | %-12.2f\n",
               i + 1,
               arr[i].date,
               arr[i].steps,
               arr[i].calories,
               arr[i].sleep_hours);
    }
}

static void show_summary(const HealthEntry *arr, size_t count) {
    if (count == 0) {
        printf("No data to summarize.\n");
        return;
    }
    long total_steps = 0;
```

```
    long total_cal = 0;
    float total_sleep = 0.0f;
    int max_steps = arr[0].steps;
    int max_cal = arr[0].calories;
    float max_sleep = arr[0].sleep_hours;

    for (size_t i = 0; i < count; ++i) {
        total_steps += arr[i].steps;
        total_cal += arr[i].calories;
        total_sleep += arr[i].sleep_hours;
        if (arr[i].steps > max_steps) max_steps = arr[i].steps;
        if (arr[i].calories > max_cal) max_cal = arr[i].calories;
        if (arr[i].sleep_hours > max_sleep) max_sleep = arr[i].sleep_hours;
    }

    printf("\nSummary for %zu day(s):\n", count);
    printf("  Total steps   : %ld\n", total_steps);
    printf("  Average steps : %.2f\n", (double) total_steps / count);
    printf("  Total calories: %ld\n", total_cal);
    printf("  Average cal   : %.2f\n", (double) total_cal / count);
    printf("  Total sleep   : %.2f hours\n", total_sleep);
    printf("  Average sleep : %.2f hours/night\n", total_sleep / count);
    printf("  Max steps     : %d\n", max_steps);
    printf("  Max calories  : %d\n", max_cal);
    printf("  Max sleep     : %.2f\n", max_sleep);
}

static void visualize_metric(const HealthEntry *arr, size_t count) {
    if (count == 0) {
        printf("No data to visualize.\n");
        return;
    }
    char buf[INPUT_BUF];
    int choice;

    printf("Choose metric to visualize:\n");
    printf("  1) Steps\n");
    printf("  2) Calories\n");
    printf("  3) Sleep hours\n");
    printf("Enter choice: ");
    if (!read_line(buf, sizeof(buf))) {
        printf("Input error.\n");
        return;
    }
```

```
    if (!parse_int(buf, &choice) || choice < 1 || choice > 3) {
        printf("Invalid choice.\n");
        return;
    }

    /* determine max value for scaling */
    double max_val = 0.0;
    for (size_t i = 0; i < count; ++i) {
        double v = 0.0;
        if (choice == 1) v = arr[i].steps;
        else if (choice == 2) v = arr[i].calories;
        else v = arr[i].sleep_hours;

        if (v > max_val) max_val = v;
    }
    if (max_val <= 0.0) {
        printf("All values are zero; nothing to visualize.\n");
        return;
    }

    const int max_bar = 30;
    printf("\nSimple bar chart:\n");
    for (size_t i = 0; i < count; ++i) {
        double v = 0.0;
        if (choice == 1) v = arr[i].steps;
        else if (choice == 2) v = arr[i].calories;
        else v = arr[i].sleep_hours;

        int bar_len = (int)((v / max_val) * max_bar + 0.5);
        if (bar_len < 0) bar_len = 0;
        if (bar_len > max_bar) bar_len = max_bar;

        printf("%s | ", arr[i].date);
        for (int j = 0; j < bar_len; ++j) putchar('#');

        if (choice == 1)
            printf(" (%.0f steps)\n", v);
        else if (choice == 2)
            printf(" (%.0f cal)\n", v);
        else
            printf(" (%.2f h)\n", v);
    }
}
```

```
int main(void) {
    HealthEntry *entries = NULL;
    size_t count = 0, cap = 0;
    char buf[INPUT_BUF];

    printf("=== Personal Health Metrics Tracker ===\n");

    for (;;) {
        printf("\nMenu:\n");
        printf("  1) Add daily entry\n");
        printf("  2) View all entries\n");
        printf("  3) Show summary\n");
        printf("  4) Visualize metric\n");
        printf("  5) Exit\n");
        printf("Choose an option: ");

        if (!read_line(buf, sizeof(buf))) {
            printf("Input error or EOF. Exiting.\n");
            break;
        }

        int choice;
        if (!parse_int(buf, &choice)) {
            printf("Please enter a number between 1 and 5.\n");
            continue;
        }

        switch (choice) {
            case 1:
                add_entry(&entries, &count, &cap);
                break;
            case 2:
                list_entries(entries, count);
                break;
            case 3:
                show_summary(entries, count);
                break;
            case 4:
                visualize_metric(entries, count);
                break;
            case 5:
                printf("Goodbye!\n");
                free(entries);
                return 0;
```

```
            default:
                printf("Invalid option. Choose 1-5.\n");
                break;
        }
    }

    free(entries);
    return 0;
}
```

OUTPUT: -