P1 報告

1. Title and Author

- Assignment number - **p1**
- Your name - **潘煒翔**
- Student number - **112950009**
- Email address - **panweikyle.sc12@nycu.edu.tw**

---

2. Description of the Problem

此程式分為三個部分：

1. 將輸入的日期轉換成對應的「星期幾」、「英文月份」、「年與日」
2. 計算兩個日期之間相差的天數
3. 由某日期加上 (或減去) 指定的天數，輸出新的日期

在本次 p1 作業中，除了上述功能之外，我也延續 p0 作業所完成的輸入檢查機制，能夠排除非法輸入，例如：

- 格式錯誤（例如缺少斜線或數字不足）
- 不存在的月份（如 13 月）
- 不合法的日期（如 32 日）
- 閏年判斷錯誤（如 2/29 但非閏年）

這些前置處理大幅提升了 p1 作業的完整性，使得程式在執行三種模式時能即時回報錯誤原因。

---

3. Main Results

(a) Program Design

將日期映射為序列天數 (serial day number)，再透過整數運算來完成各種操作
如下：

1. **輸入與檢查**：`parseYMD` 負責解析字串並檢查錯誤
2. **標準化格式**：`normalizeYMD` 產生固定格式 `yyyy/mm/dd`，避免多種表示造成混淆
3. **核心運算**：

    - `toSerial`：日期 → 累計天數
    - `fromSerial`：累計天數 → 日期
    - `dayOfWeek`：計算星期幾
    - `dateDiffDays`：計算兩日期相差天數
    - `dateAddDays`：日期加減 n 天
4. **互動主程式**：
    - a / 1：顯示某日期的星期幾
    - b / 2：顯示兩日期相差天數
    - c / 3：計算加減 n 天後的新日期

程式碼片段：

- `parseYMD`
- `toSerial` & `fromSerial`

(b) Data Structures

程式中主要使用的資料結構：

1. **`struct Date { int y, m, d; }`**
    - 將日期三元素 (年、月、日) 封裝，方便函式傳遞
2. **靜態陣列** **`base[13]`**
    - 快速查表各月份天數，並搭配 `isLeap` 修正閏年二月
3. **自訂字串函數**
    - `my_strlen`, `my_strcpy`, `my_trim`, `my_isdigit`
    - 避免使用標準函式庫，完全掌控字串處理邏輯
4. **字串表**
    - `MONTH[13]`、`WEEKNAME[7]` 儲存英文月份與星期名稱。
    - 配合輸出格式，滿足作業需求

程式碼片段：

- `struct Date`
- `daysInMonth`
- `isLeap`
- `MONTH`
- `WEEKNAME`

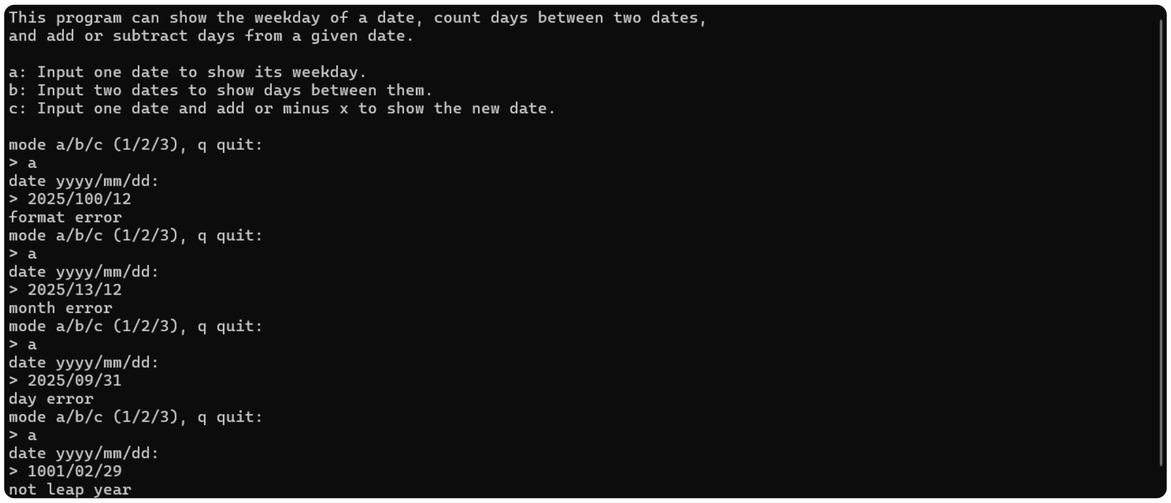
© Program Listing with Comments

完整程式碼附於附錄，主體中僅節錄關鍵部分
程式已加入必要註解，說明函數用途與錯誤代碼

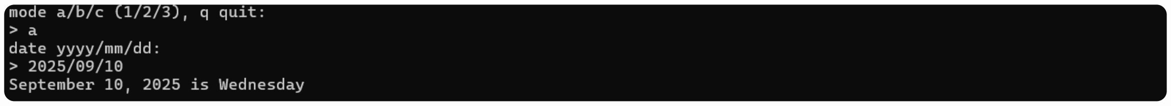
(d) Program Outputs

程式可正確處理三種輸入模式，並能即時回報錯誤：

- **模式 a**：輸入 `2019/9/20` → 輸出 `September 20, 2019 is Friday`

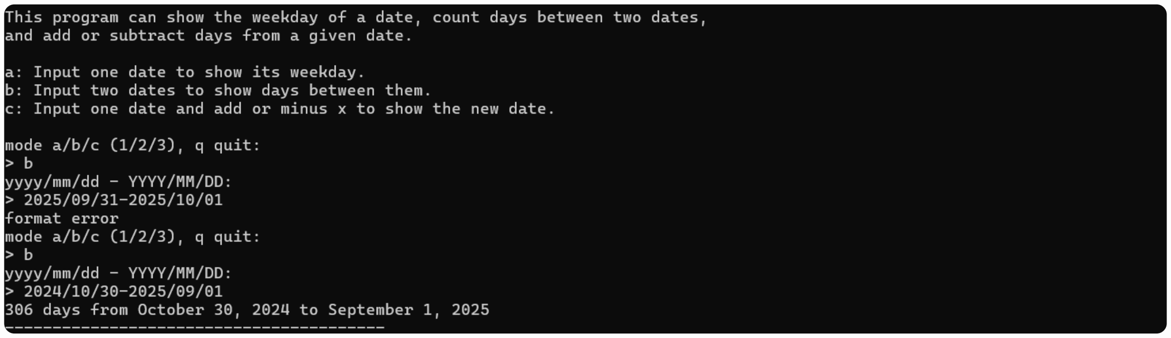


```
This program can show the weekday of a date, count days between two dates,
and add or subtract days from a given date.

a: Input one date to show its weekday.
b: Input two dates to show days between them.
c: Input one date and add or minus x to show the new date.

mode a/b/c (1/2/3), q quit:
> a
date yyyy/mm/dd:
> 2025/100/12
format error
mode a/b/c (1/2/3), q quit:
> a
date yyyy/mm/dd:
> 2025/13/12
month error
mode a/b/c (1/2/3), q quit:
> a
date yyyy/mm/dd:
> 2025/09/31
day error
mode a/b/c (1/2/3), q quit:
> a
date yyyy/mm/dd:
> 1001/02/29
not leap year
```



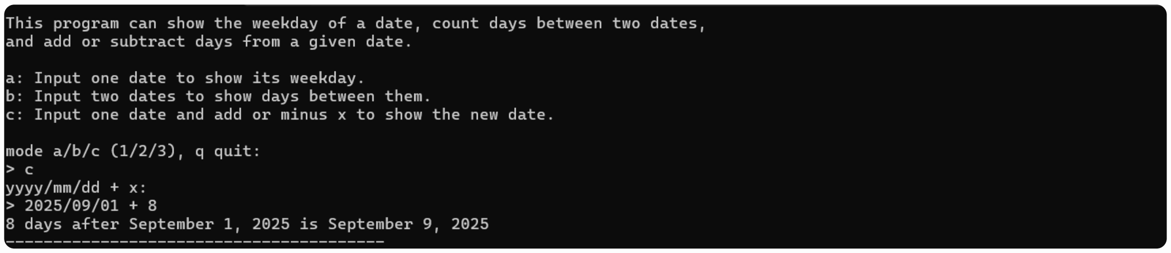
```
mode a/b/c (1/2/3), q quit:
> a
date yyyy/mm/dd:
> 2025/09/10
September 10, 2025 is Wednesday
```

- **模式 b**：輸入 `2018/9/20 - 2019/9/20` → 輸出 `365 days from September 20, 2018 to September 20, 2019`



```
This program can show the weekday of a date, count days between two dates,
and add or subtract days from a given date.

a: Input one date to show its weekday.
b: Input two dates to show days between them.
c: Input one date and add or minus x to show the new date.

mode a/b/c (1/2/3), q quit:
> b
yyyy/mm/dd - YYYY/MM/DD:
> 2025/09/31-2025/10/01
format error
mode a/b/c (1/2/3), q quit:
> b
yyyy/mm/dd - YYYY/MM/DD:
> 2024/10/30-2025/09/01
306 days from October 30, 2024 to September 1, 2025
----------------------------------------
```

- **模式 c**：輸入 `2019/9/20 + 365` → 輸出 `365 days after September 20, 2019 is September 19, 2020`



```
This program can show the weekday of a date, count days between two dates,
and add or subtract days from a given date.

a: Input one date to show its weekday.
b: Input two dates to show days between them.
c: Input one date and add or minus x to show the new date.

mode a/b/c (1/2/3), q quit:
> c
yyyy/mm/dd + x:
> 2025/09/01 + 8
8 days after September 1, 2025 is September 9, 2025
----------------------------------------
```

---

4. Performance Evaluation

(a)

n (triplets)	total_ops	elapsed_seconds
100	300	0.001263492
200	600	0.002456190
300	900	0.003578381
400	1200	0.004773190
500	1500	0.005999574
600	1800	0.007164831
700	2100	0.008261547
800	2400	0.009790533
900	2700	0.027765851
1000	3000	0.015691816

(b)

time_limit	max_n_triplets (est.)	approx_total_ops
1 minute	3,823,649	11,470,947
5 minutes	19,118,246	57,354,738
10 minutes	38,236,492	114,709,476

---

5. Conclusions

一開始在做 p0 的時候，我就先想要把輸入的日期格式弄清楚，判斷對錯，尤其是格式不正確的情況。那時候我寫 code == 1 來處理格式錯誤，但一測就發現有很多沒考慮到的例子，例如年分不夠

四位數、分隔符號錯了、輸入最後多了一些空格之類的。這些問題讓我卡了一陣子，後來和同學討論過，再加上一些網路查到的方法，我才把條件補齊。

接著，我又遇到月份顯示一直錯的問題。原來是因為我把一月直接放在陣列的索引 0，結果整個都偏掉了。最後我乾脆把索引 0 空出來，從索引 1 才開始放「January」，才讓對應正確。後來還有個麻煩是 02 和 2 這種輸入會搞混，所以我自己寫了一個 normalizeYMD，統一把日期轉成 yyyy/mm/dd 的固定格式。這樣在後面判斷時就不會再出現亂掉的情況。

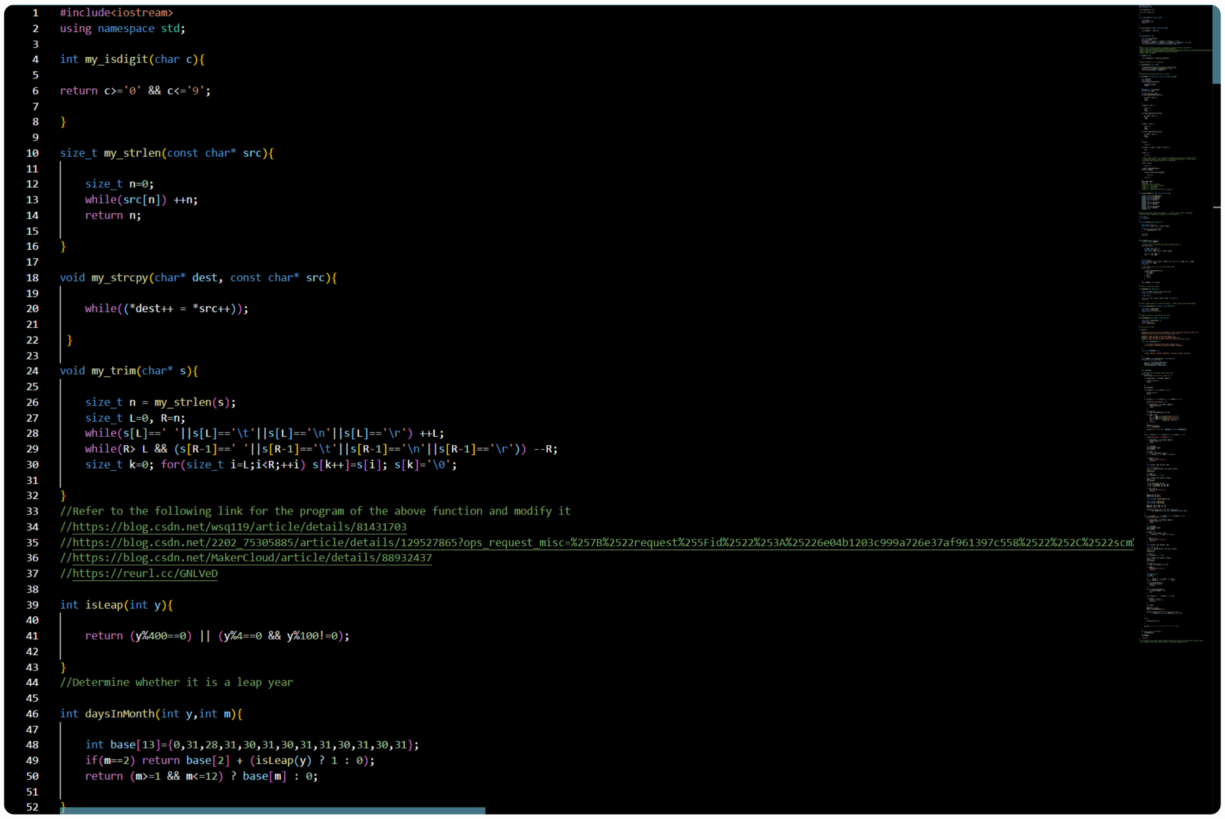
在資料結構方面，我覺得最關鍵的就是我設計了一個 struct Date { int y, m, d; }，把年、月、日包在一起，這樣不管是要傳進函數還是要回傳結果，都比較乾淨。再加上我用一個 base[13] 陣列來存每個月的天數，搭配 isLeap 去修正二月，基本上就能處理各種月份的狀況。這些小小的結構設計，後來證明很方便。

等到 p1 作業更新，規格要求支援 a/b/c 三種模式，我一開始覺得很複雜，不過後來想到把日期轉成「連續天數」來做就簡單很多。先用 toSerial 把日期變成整數，然後所有加減差都在整數上完成，最後再用 fromSerial 轉回日期。這樣 dayOfWeek、dateDiffDays、dateAddDays 全部都可以統一處理，程式的邏輯也變得很清楚。

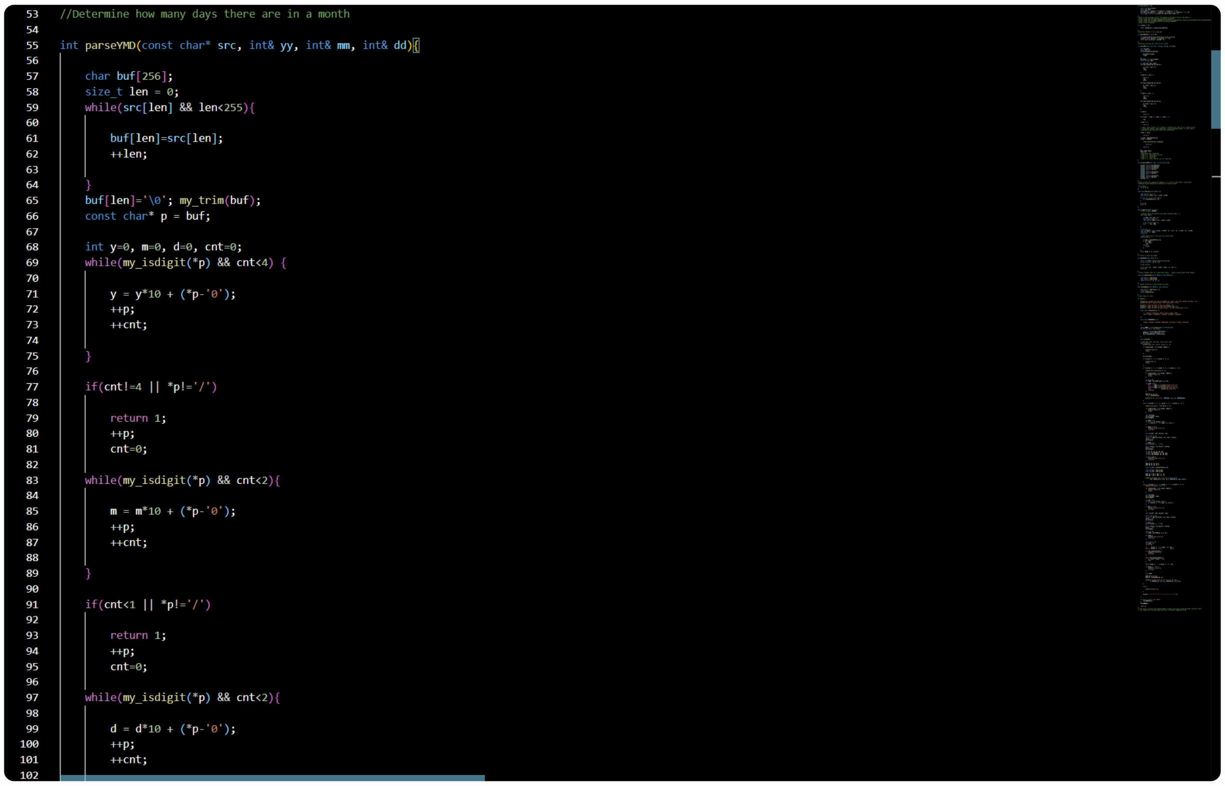
另外，字串處理的部分一開始我完全沒想到會卡這麼多。後來才知道 strcpy 很好用，但我還是自己寫了 my_strlen、my_strcpy、my_trim、my_isdigit，一方面是符合作業要求，另一方面也是自己掌握細節。尤其是 my_trim，讓我可以把前後空白都處理掉，不然輸入如果多一點空格，整個判斷就會失敗。

總結來說，這次作業最大的收穫是我真的體會到「資料結構設計」的重要性。從一開始處理各種格式錯誤，到後來用 struct Date 和「天數序列化」的概念，我學到只要基礎設計穩，後面要加新功能就不會太痛苦。

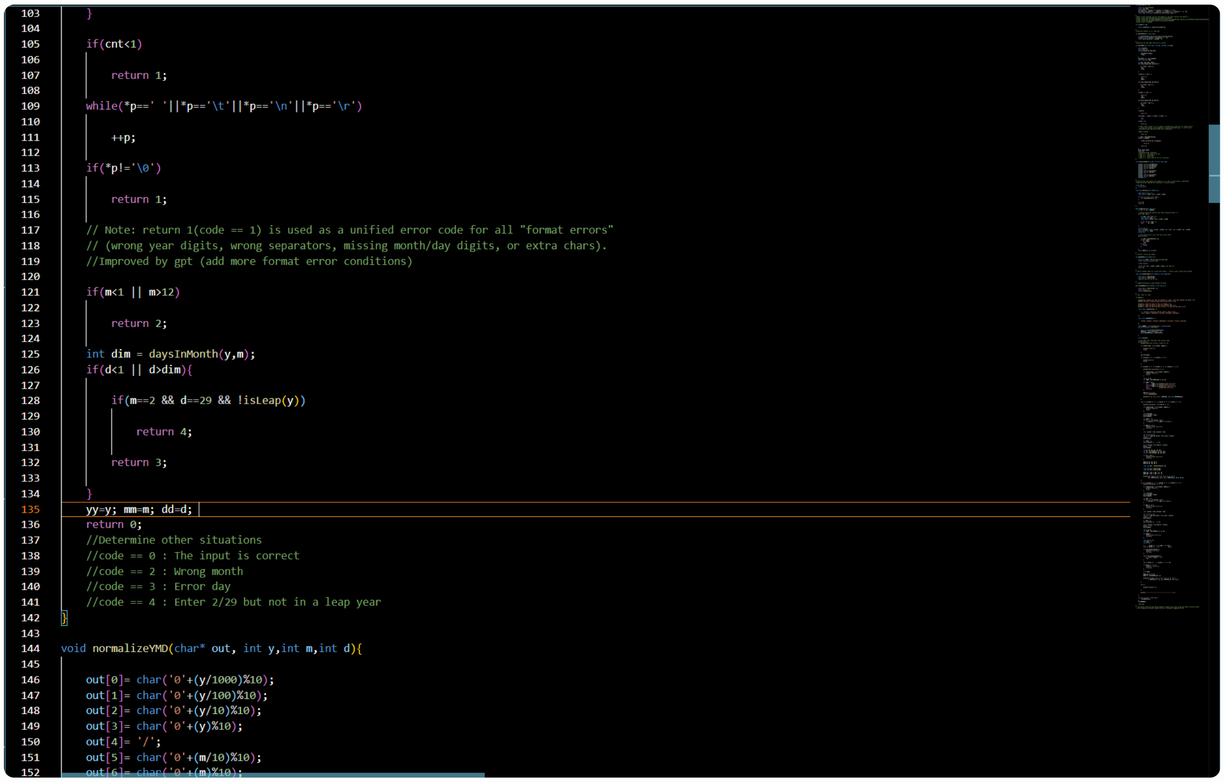
6. 附錄



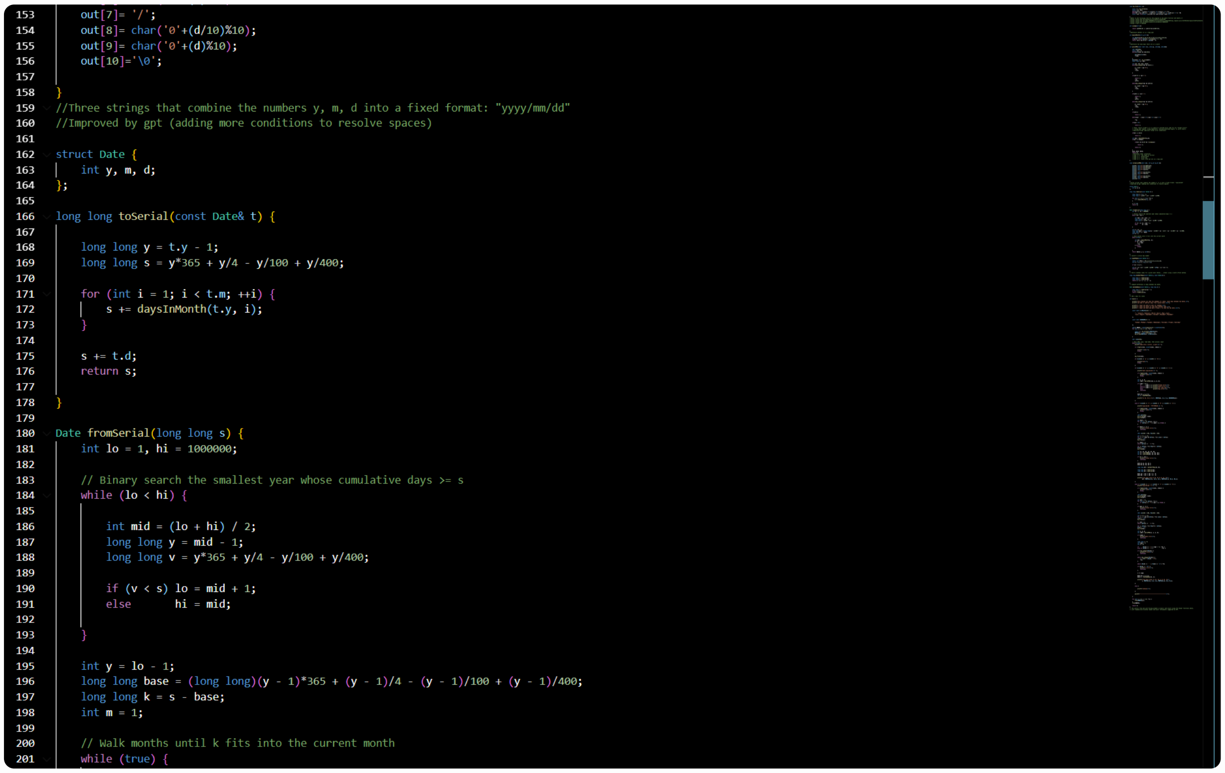
```cpp
#include<iostream>
using namespace std;

int my_isdigit(char c){

return c>='0' && c<='9';

}

size_t my_strlen(const char* src){

    size_t n=0;
    while(src[n]) ++n;
    return n;

}

void my_strcpy(char* dest, const char* src){

    while((*dest++ = *src++));

 }

void my_trim(char* s){

    size_t n = my_strlen(s);
    size_t L=0, R=n;
    while(s[L]==' '||s[L]=='\t'||s[L]=='\n'||s[L]=='\r') ++L;
    while(R> L && (s[R-1]==' '||s[R-1]=='\t'||s[R-1]=='\n'||s[R-1]=='\r')) --R;
    size_t k=0; for(size_t i=L;i<R;++i) s[k++]=s[i]; s[k]='\0';

}
//Refer to the following link for the program of the above function and modify it
//https://blog.csdn.net/wsq119/article/details/81431703
//https://blog.csdn.net/2202_75305885/article/details/129527865?ops_request_misc=%257B%2522request%255Fid%2522%253A%25226e04b1203c999a726e37af961397c558%2522%252C%2522scm
//https://blog.csdn.net/MakerCloud/article/details/88932437
//https://reurl.cc/GNLVeD

int isLeap(int y){

    return (y%400==0) || (y%4==0 && y%100!=0);

}
//Determine whether it is a leap year

int daysInMonth(int y,int m){

    int base[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    if(m==2) return base[2] + (isLeap(y) ? 1 : 0);
    return (m>=1 && m<=12) ? base[m] : 0;

}
```



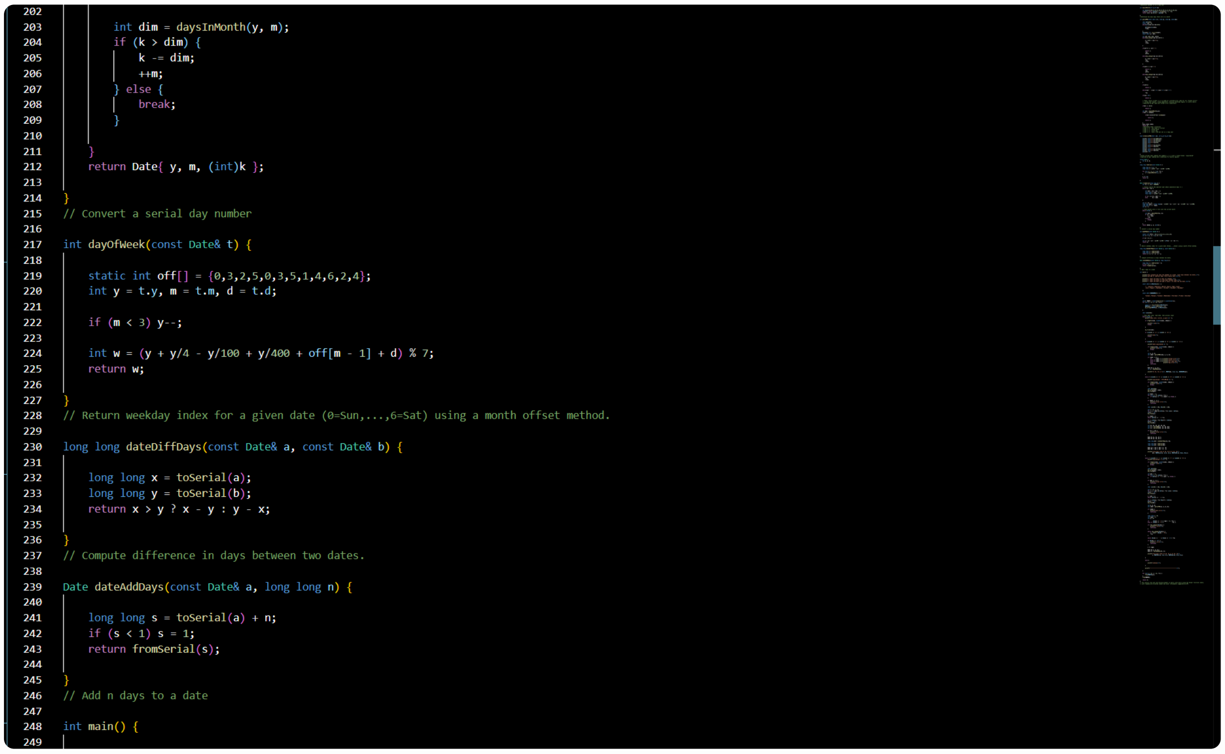
```cpp
//Determine how many days there are in a month

int parseYMD(const char* src, int& yy, int& mm, int& dd){

    char buf[256];
    size_t len = 0;
    while(src[len] && len<255){

        buf[len]=src[len];
        ++len;

    }
    buf[len]='\0'; my_trim(buf);
    const char* p = buf;

    int y=0, m=0, d=0, cnt=0;
    while(my_isdigit(*p) && cnt<4) {

        y = y*10 + (*p-'0');
        ++p;
        ++cnt;

    }

    if(cnt!=4 || *p!='/')

        return 1;
        ++p;
        cnt=0;

    while(my_isdigit(*p) && cnt<2){

        m = m*10 + (*p-'0');
        ++p;
        ++cnt;

    }

    if(cnt<1 || *p!='/')

        return 1;
        ++p;
        cnt=0;

    while(my_isdigit(*p) && cnt<2){

        d = d*10 + (*p-'0');
        ++p;
        ++cnt;

```



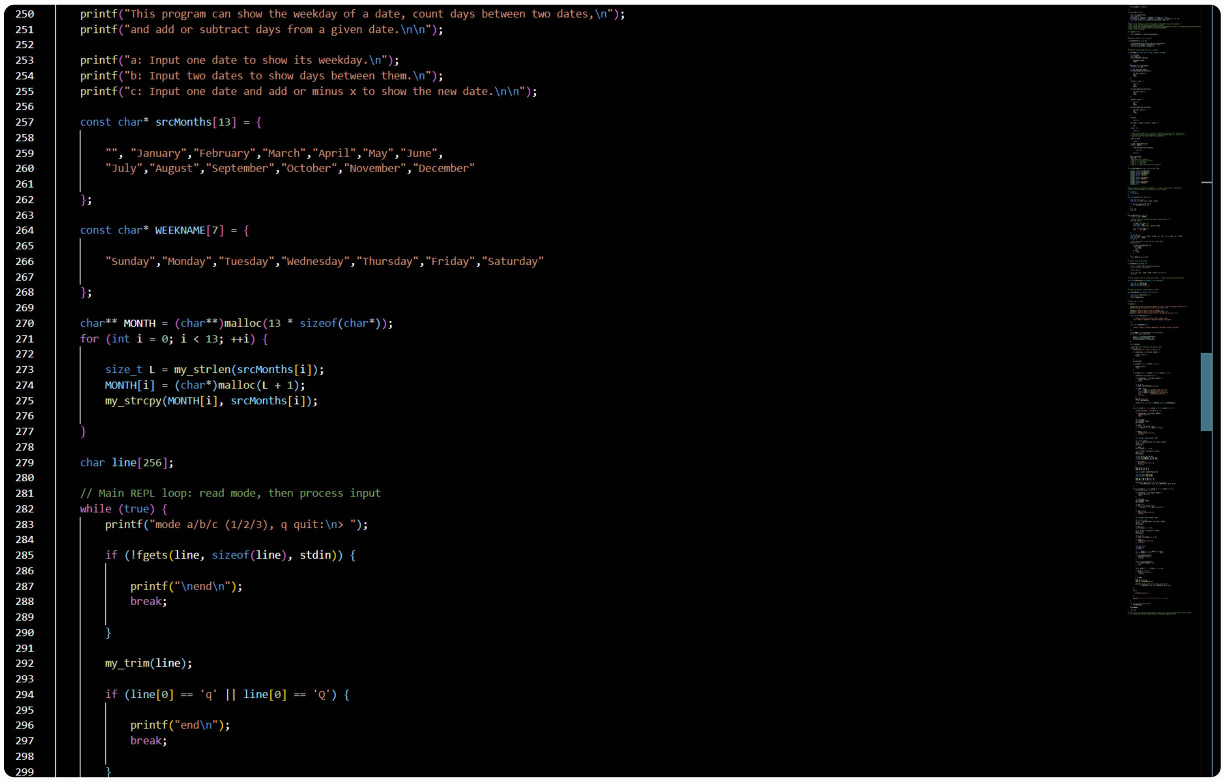
```cpp
    }

    if(cnt<1)

        return 1;

    while(*p==' '||*p=='\t'||*p=='\n'||*p=='\r')

        ++p;

    if(*p!='\0')

        return 1;

    // Note: return 1(code == 1) is used as a unified error code for all "format errors"
    // (wrong year digits, wrong separators, missing month/day digits, or extra chars).
    //Improved by gpt (add more format error conditions)

    if(m<1 || m>12)

        return 2;

    int dim = daysInMonth(y,m);
    if(d<1 || d>dim){

        if(m==2 && d==29 && !isLeap(y))

            return 4;

        return 3;

    }
    yy=y; mm=m; dd=d;
    return 0;
    //Determine other situations
    //code == 0 : The input is correct
    //code == 2 : Wrong month
    //code == 3 : Error day
    //code == 4 : Enter 2/29 but not in a leap year
}

void normalizeYMD(char* out, int y,int m,int d){

    out[0]= char('0'+(y/1000)%10);
    out[1]= char('0'+(y/100)%10);
    out[2]= char('0'+(y/10)%10);
    out[3]= char('0'+(y)%10);
    out[4]= '/';
    out[5]= char('0'+(m/10)%10);
    out[6]= char('0'+(m)%10);
```



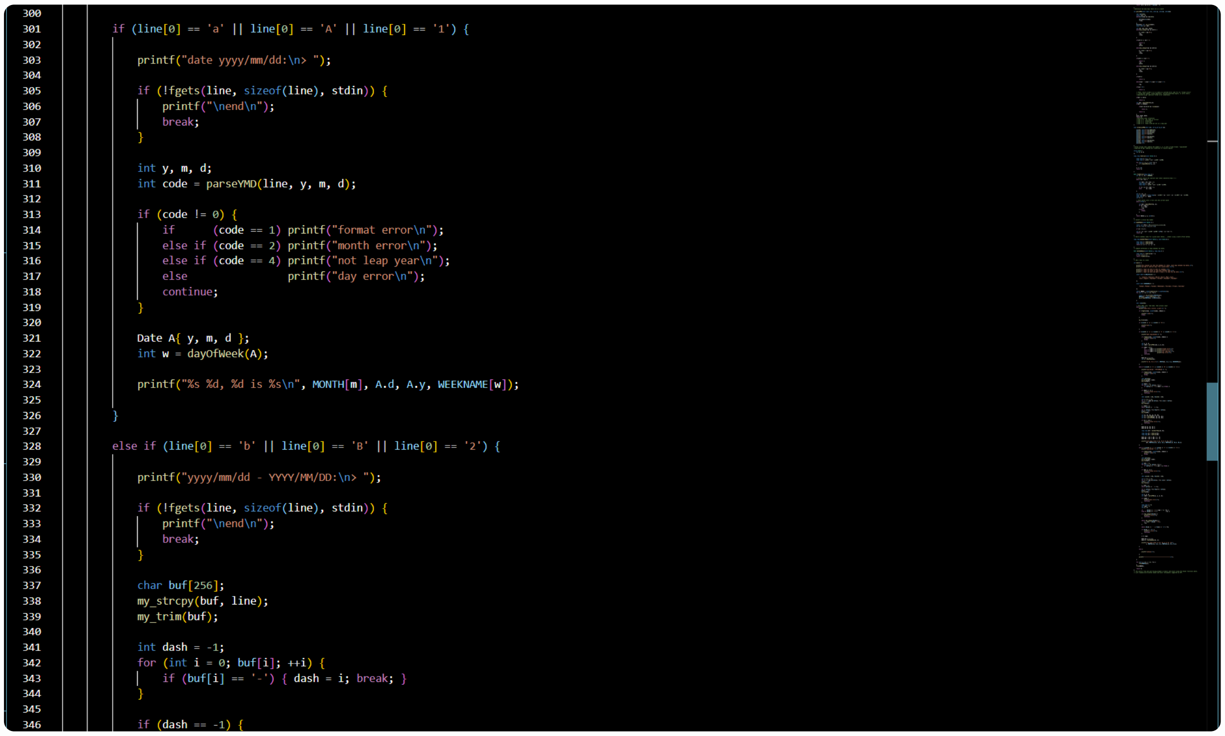
```cpp
    out[7]= '/';
    out[8]= char('0'+(d/10)%10);
    out[9]= char('0'+(d)%10);
    out[10]='\0';

}
//Three strings that combine the numbers y, m, d into a fixed format: "yyyy/mm/dd"
//Improved by gpt (adding more conditions to resolve spaces)

struct Date {
    int y, m, d;
};

long long toSerial(const Date& t) {

    long long y = t.y - 1;
    long long s = y*365 + y/4 - y/100 + y/400;

    for (int i = 1; i < t.m; ++i) {
        s += daysInMonth(t.y, i);
    }

    s += t.d;
    return s;

}

Date fromSerial(long long s) {
    int lo = 1, hi = 1000000;

    // Binary search the smallest year whose cumulative days >= s
    while (lo < hi) {

        int mid = (lo + hi) / 2;
        long long y = mid - 1;
        long long v = y*365 + y/4 - y/100 + y/400;

        if (v < s) lo = mid + 1;
        else       hi = mid;

    }

    int y = lo - 1;
    long long base = (long long)(y - 1)*365 + (y - 1)/4 - (y - 1)/100 + (y - 1)/400;
    long long k = s - base;
    int m = 1;

    // Walk months until k fits into the current month
    while (true) {
```



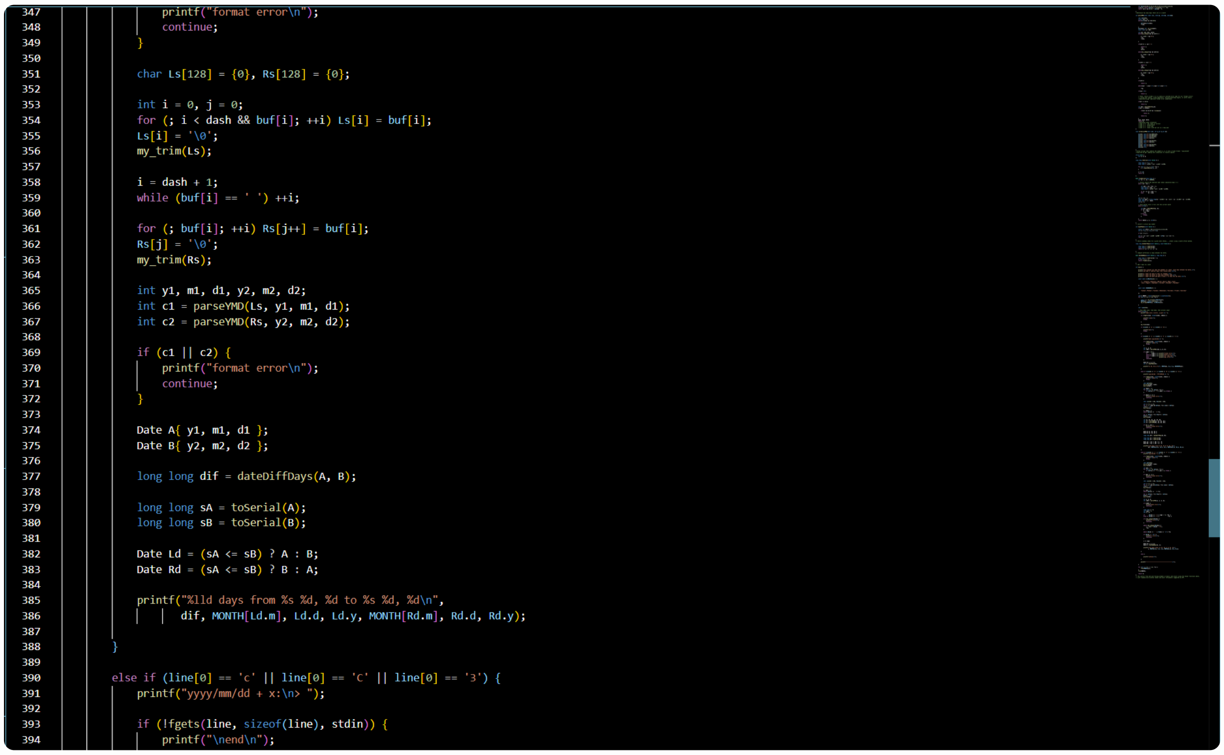
```cpp

        int dim = daysInMonth(y, m);
        if (k > dim) {
            k -= dim;
            ++m;
        } else {
            break;
        }

    }
    return Date{ y, m, (int)k };

}
// Convert a serial day number

int dayOfWeek(const Date& t) {

    static int off[] = {0,3,2,5,0,3,5,1,4,6,2,4};
    int y = t.y, m = t.m, d = t.d;

    if (m < 3) y--;

    int w = (y + y/4 - y/100 + y/400 + off[m - 1] + d) % 7;
    return w;

}
// Return weekday index for a given date (0=Sun,...,6=Sat) using a month offset method.

long long dateDiffDays(const Date& a, const Date& b) {

    long long x = toSerial(a);
    long long y = toSerial(b);
    return x > y ? x - y : y - x;

}
// Compute difference in days between two dates.

Date dateAddDays(const Date& a, long long n) {

    long long s = toSerial(a) + n;
    if (s < 1) s = 1;
    return fromSerial(s);

}
// Add n days to a date

int main() {

```



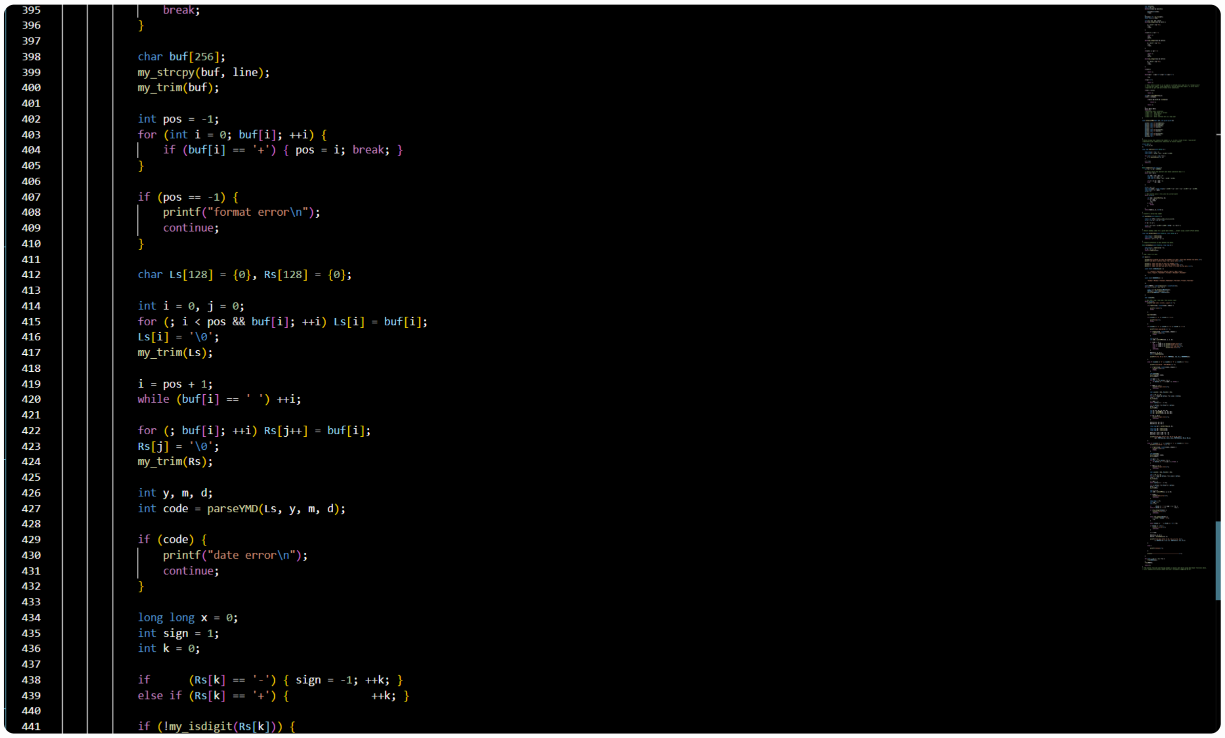
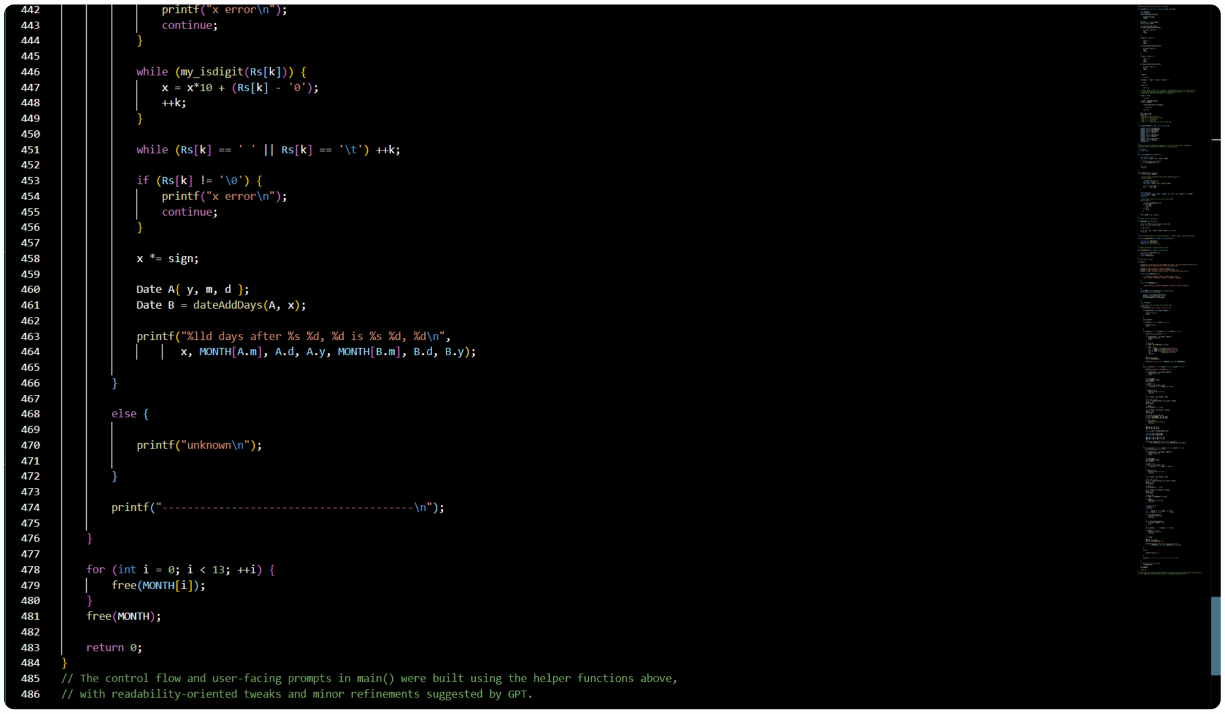
```cpp
    printf("This program can show the weekday of a date, count days between two dates,\n");
    printf("and add or subtract days from a given date.\n\n");

    printf("a: Input one date to show its weekday.\n");
    printf("b: Input two dates to show days between them.\n");
    printf("c: Input one date and add or minus x to show the new date.\n\n");

    const char* srcMonths[13] = {

        "", "January","February","March","April","May","June",
        "July","August","September","October","November","December"

    };

    const char* WEEKNAME[7] = {

        "Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"

    };

    char** MONTH = (char**)malloc(13 * sizeof(char*));
    for (int i = 0; i < 13; ++i) {

        size_t L = my_strlen(srcMonths[i]);
        MONTH[i] = (char*)malloc(L + 1);
        my_strcpy(MONTH[i], srcMonths[i]);

    }

    char line[256];

    // Main REPL loop: read mode, then process input
    while (true) {
        printf("mode a/b/c (1/2/3), q quit:\n> ");

        if (!fgets(line, sizeof(line), stdin)) {

            printf("\nend\n");
            break;

        }

        my_trim(line);

        if (line[0] == 'q' || line[0] == 'Q') {

            printf("end\n");
            break;

        }
```



```c

        if (line[0] == 'a' || line[0] == 'A' || line[0] == '1') {

            printf("date yyyy/mm/dd:\n> ");

            if (!fgets(line, sizeof(line), stdin)) {
                printf("\nend\n");
                break;
            }

            int y, m, d;
            int code = parseYMD(line, y, m, d);

            if (code != 0) {
                if      (code == 1) printf("format error\n");
                else if (code == 2) printf("month error\n");
                else if (code == 4) printf("not leap year\n");
                else                printf("day error\n");
                continue;
            }

            Date A{ y, m, d };
            int w = dayOfWeek(A);

            printf("%s %d, %d is %s\n", MONTH[m], A.d, A.y, WEEKNAME[w]);

        }

        else if (line[0] == 'b' || line[0] == 'B' || line[0] == '2') {

            printf("yyyy/mm/dd - YYYY/MM/DD:\n> ");

            if (!fgets(line, sizeof(line), stdin)) {
                printf("\nend\n");
                break;
            }

            char buf[256];
            my_strcpy(buf, line);
            my_trim(buf);

            int dash = -1;
            for (int i = 0; buf[i]; ++i) {
                if (buf[i] == '-') { dash = i; break; }
            }

            if (dash == -1) {
```



```c
                printf("format error\n");
                continue;
            }

            char Ls[128] = {0}, Rs[128] = {0};

            int i = 0, j = 0;
            for (; i < dash && buf[i]; ++i) Ls[i] = buf[i];
            Ls[i] = '\0';
            my_trim(Ls);

            i = dash + 1;
            while (buf[i] == ' ') ++i;

            for (; buf[i]; ++i) Rs[j++] = buf[i];
            Rs[j] = '\0';
            my_trim(Rs);

            int y1, m1, d1, y2, m2, d2;
            int c1 = parseYMD(Ls, y1, m1, d1);
            int c2 = parseYMD(Rs, y2, m2, d2);

            if (c1 || c2) {
                printf("format error\n");
                continue;
            }

            Date A{ y1, m1, d1 };
            Date B{ y2, m2, d2 };

            long long dif = dateDiffDays(A, B);

            long long sA = toSerial(A);
            long long sB = toSerial(B);

            Date Ld = (sA <= sB) ? A : B;
            Date Rd = (sA <= sB) ? B : A;

            printf("%lld days from %s %d, %d to %s %d, %d\n",
                   dif, MONTH[Ld.m], Ld.d, Ld.y, MONTH[Rd.m], Rd.d, Rd.y);

        }

        else if (line[0] == 'c' || line[0] == 'C' || line[0] == '3') {
            printf("yyyy/mm/dd + x:\n> ");

            if (!fgets(line, sizeof(line), stdin)) {
                printf("\nend\n");
```


```c
                break;
            }

            char buf[256];
            my_strcpy(buf, line);
            my_trim(buf);

            int pos = -1;
            for (int i = 0; buf[i]; ++i) {
                if (buf[i] == '+') { pos = i; break; }
            }

            if (pos == -1) {
                printf("format error\n");
                continue;
            }

            char Ls[128] = {0}, Rs[128] = {0};

            int i = 0, j = 0;
            for (; i < pos && buf[i]; ++i) Ls[i] = buf[i];
            Ls[i] = '\0';
            my_trim(Ls);

            i = pos + 1;
            while (buf[i] == ' ') ++i;

            for (; buf[i]; ++i) Rs[j++] = buf[i];
            Rs[j] = '\0';
            my_trim(Rs);

            int y, m, d;
            int code = parseYMD(Ls, y, m, d);

            if (code) {
                printf("date error\n");
                continue;
            }

            long long x = 0;
            int sign = 1;
            int k = 0;

            if      (Rs[k] == '-') { sign = -1; ++k; }
            else if (Rs[k] == '+') {             ++k; }

            if (!my_isdigit(Rs[k])) {
```

```c
                printf("x error\n");
                continue;
            }

            while (my_isdigit(Rs[k])) {
                x = x*10 + (Rs[k] - '0');
                ++k;
            }

            while (Rs[k] == ' ' || Rs[k] == '\t') ++k;

            if (Rs[k] != '\0') {
                printf("x error\n");
                continue;
            }

            x *= sign;

            Date A{ y, m, d };
            Date B = dateAddDays(A, x);

            printf("%lld days after %s %d, %d is %s %d, %d\n",
                   x, MONTH[A.m], A.d, A.y, MONTH[B.m], B.d, B.y);

        }

        else {

            printf("unknown\n");

        }

        printf("----------------------------------------\n");

    }

    for (int i = 0; i < 13; ++i) {
        free(MONTH[i]);
    }
    free(MONTH);

    return 0;
}
// The control flow and user-facing prompts in main() were built using the helper functions above,
// with readability-oriented tweaks and minor refinements suggested by GPT.
```