Experiment8-董皓彧

环境：

```
gcc.exe (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
Visual Stdio Code 1.83.1
```

作业仓库地址：
https://github.com/FHYQ-Dong/Tsinghua-Program-Design-Assignments/tree/main/Experiment8

必做题

Experiment8-1

题目：

> 39 个犹太人与约瑟夫躲到山洞，39 个犹太人决定宁死也不要被敌人抓到，于是决定了自杀方式：40 个人排成一个圆圈，由第 1 个人开始报数，每报到第 3 人该人就必须自杀，然后再由下一个重新报数，直到所有人都自杀身亡为止。然而约瑟夫并不想死，他站在某个位置上，最终逃过了这场死亡游戏。试问：这个位置是哪个编号？（说明：完成此题时，编号规定从0开始，即 40 个人的编号为 0-39）

输入格式：

```
无
```

输出格式：

```
四行：
前两行：递归解法：第一行为答案，第二行为耗时；
后两行：循环解法：第三行为答案，第四行为耗时
```

代码：

```c
#include <stdio.h>
#include <Windows.h>

typedef struct Node {
    int idx;
    struct Node *next;
} Node;
typedef struct Loop {
    Node *head, *tail, *cur;
    int size;
    void (*init)(struct Loop *this, int _size);
} Loop;

double run_time;
LARGE_INTEGER time_start, time_over;    //开始时间，结束时间
double dqFreq;      //计时器频率
```

```
LARGE_INTEGER f;
int cnt = 40, ans = 0;

void _init_loop(Loop *this, int _size) {
    this->head = NULL;
    this->tail = NULL;
    this->cur = (Node*)malloc(sizeof(Node));
    this->size = _size;
    for(int i=this->size-1; i>=0; --i) {
        Node *tmp = (Node*)malloc(sizeof(Node));
        tmp->idx = i;
        tmp->next = this->head;
        this->head = tmp;
        if(i == this->size-1) this->tail = tmp;
        this->tail->next = this->head;
    }
    this->cur = this->head;
    return;
}

void get_ans_loop(Loop *people, int cnt, int *ans) {
    while(people->size > 1) {
        people->cur = people->cur->next;
        Node *tmp = people->cur->next;
        people->cur->next = tmp->next;
        free(tmp);
        people->cur = people->cur->next;
        --people->size;
    }
    *ans = people->cur->idx;
    free(people->cur);
    return;
}

void get_ans_recurrence(Loop *people, int cur_alive, int *ans) {
    if (cur_alive == 1) {
        *ans = people->cur->idx;
        free(people->cur);
        return;
    }
    people->cur = people->cur->next;
    Node *tmp = people->cur->next;
    people->cur->next = tmp->next;
    free(tmp);
    people->cur = people->cur->next;
    --people->size;
    get_ans_recurrence(people, cur_alive-1, ans);
    return;
}

void timer_init() {
    QueryPerformanceFrequency(&f);
    dqFreq=(double)f.QuadPart;
    QueryPerformanceCounter(&time_start);
}
```

```
int main() {
    Loop *people = (Loop*)malloc(sizeof(Loop));
    people->init = _init_loop;
    timer_init();

    people->init(people, cnt);
    QueryPerformanceCounter(&time_start);   //计时开始
    get_ans_loop(people, cnt, &ans);
    QueryPerformanceCounter(&time_over);    //计时结束
    printf("Answer_loop: %d\n", ans);
    run_time=1000*(time_over.QuadPart-time_start.QuadPart)/dqFreq;
    printf("Time_loop: %lfms\n", run_time);

    people->init(people, cnt);
    QueryPerformanceCounter(&time_start);   //计时开始
    get_ans_recurrence(people, cnt, &ans);
    QueryPerformanceCounter(&time_over);    //计时结束
    printf("Answer_recurrence: %d\n", ans);
    run_time=1000*(time_over.QuadPart-time_start.QuadPart)/dqFreq;
    printf("Time_recurrence: %lfms\n", run_time);
    return 0;
}
```

输入1：

```

```

输出1：

```
Answer_loop: 27
Time_loop: 0.001500ms
Answer_recurrence: 27
Time_recurrence: 0.000900ms
```

Experiment8-2

题目：

> 从楼上走到楼下共有 n 级台阶，每一步有三种走法，走一级、走两级或走三级台阶。问恰好走完这n级台阶共有多少种不同的方案。

输入格式：

> 一行，一个整数，为台阶数 n

输出格式：

> 四行：
> 前两行：递归解法：第一行为答案，第二行为耗时；
> 后两行：循环解法：第三行为答案，第四行为耗时

代码：

```
#include<stdio.h>
#include<time.h>
#include<Windows.h>
#include<stdlib.h>

int ans[50+1];
int n;

double run_time;
LARGE_INTEGER time_start, time_over;    //开始时间，结束时间
double dqFreq;      //计时器频率
LARGE_INTEGER f;

int GoDown_recurrence(int x) {
    if(!ans[x]) ans[x] = GoDown_recurrence(x-1) + GoDown_recurrence(x-2) +
GoDown_recurrence(x-3);
    return ans[x];
}

int GoDown_loop(int x) {
    for(int i = 4; i <= x; i++) ans[i] = ans[i-1] + ans[i-2] + ans[i-3];
    return ans[x];
}

void timer_init() {
    QueryPerformanceFrequency(&f);
    dqFreq=(double)f.QuadPart;
    QueryPerformanceCounter(&time_start);
}

void solve_recurrence() {
    memset(ans, 0, sizeof(ans));
    ans[1] = 1; ans[2] = 2; ans[3] = 4;
    timer_init();

    QueryPerformanceCounter(&time_start);   //计时开始
    printf("Answer_recurrence: %d\n", GoDown_recurrence(n));
    QueryPerformanceCounter(&time_over);    //计时结束
    run_time=1000*(time_over.QuadPart-time_start.QuadPart)/dqFreq;
    printf("Time_recurrence: %lfms\n", run_time);
    return;
}

void solve_loop() {
    memset(ans, 0, sizeof(ans));
    ans[1] = 1; ans[2] = 2; ans[3] = 4;
    timer_init();
    QueryPerformanceCounter(&time_start);   //计时开始
    printf("Answer_loop: %d\n", GoDown_loop(n));
    QueryPerformanceCounter(&time_over);    //计时结束
    run_time=1000*(time_over.QuadPart-time_start.QuadPart)/dqFreq;
    printf("Time_loop: %lfms\n", run_time);
    return;
```

```
}

int main() {
    scanf("%d", &n);
    solve_recurrence();
    solve_loop();
    return 0;
}
```

输入1：

```
5
```

输出1：

```
Answer_recurrence: 13
Time_recurrence: 0.004000ms
Answer_loop: 13
Time_loop: 0.000300ms
```

输入2：

```
6
```

输出2：

```
Answer_recurrence: 24
Time_recurrence: 0.003900ms
Answer_loop: 24
Time_loop: 0.000300ms
```

输入3：

```
7
```

输出3：

```
Answer_recurrence: 44
Time_recurrence: 0.004500ms
Answer_loop: 44
Time_loop: 0.000200ms
```

输入4：

```
8
```

输出4：

```
Answer_recurrence: 81
Time_recurrence: 0.004000ms
Answer_loop: 81
Time_loop: 0.000200ms
```

输入5：

```
9
```

输出5：

```
Answer_recurrence: 149
Time_recurrence: 0.004000ms
Answer_loop: 149
Time_loop: 0.000200ms
```

输入6：

```
10
```

输出6：

```
Answer_recurrence: 274
Time_recurrence: 0.003500ms
Answer_loop: 274
Time_loop: 0.000200ms
```

输入7：

```
11
```

输出7：

```
Answer_recurrence: 504
Time_recurrence: 0.003800ms
Answer_loop: 504
Time_loop: 0.000200ms
```

输入8：

```
12
```

输出8：

```
Answer_recurrence: 927
Time_recurrence: 0.004000ms
Answer_loop: 927
Time_loop: 0.000300ms
```

输入9：

```
13
```

输出9：

```
Answer_recurrence: 1705
Time_recurrence: 0.003800ms
Answer_loop: 1705
Time_loop: 0.000300ms
```

输入10：

```
14
```

输出10：

```
Answer_recurrence: 3136
Time_recurrence: 0.004100ms
Answer_loop: 3136
Time_loop: 0.000400ms
```

输入11：

```
15
```

输出11：

```
Answer_recurrence: 5768
Time_recurrence: 0.003600ms
Answer_loop: 5768
Time_loop: 0.000300ms
```

输入12：

```
16
```

输出12：

```
Answer_recurrence: 10609
Time_recurrence: 0.003400ms
Answer_loop: 10609
Time_loop: 0.000200ms
```

输入13：

```
17
```

输出13：

```
Answer_recurrence: 19513
Time_recurrence: 0.003500ms
Answer_loop: 19513
Time_loop: 0.000300ms
```

输入14:

```
18
```

输出14:

```
Answer_recurrence: 35890
Time_recurrence: 0.003400ms
Answer_loop: 35890
Time_loop: 0.000300ms
```

输入15:

```
19
```

输出15:

```
Answer_recurrence: 66012
Time_recurrence: 0.003700ms
Answer_loop: 66012
Time_loop: 0.000400ms
```

输入16:

```
20
```

输出16:

```
Answer_recurrence: 121415
Time_recurrence: 0.003300ms
Answer_loop: 121415
Time_loop: 0.000300ms
```

输入17:

```
25
```

输出17:

```
Answer_recurrence: 2555757
Time_recurrence: 0.003500ms
Answer_loop: 2555757
Time_loop: 0.000300ms
```

输入18:

```
35
```

输出18：

```
Answer_recurrence: 1132436852
Time_recurrence: 0.003900ms
Answer_loop: 1132436852
Time_loop: 0.000300ms
```

选做题

Optional-Experiment8-1

题目：

> 有一群鸡和一群兔，两种动物只数相同。两种动物的脚的总数都是三位数，且这两个三位数的六个数字分别是0，1，2，3，4，5。编程求鸡和兔的只数是多少？ 它们的脚数各是多少？

输入格式：

```
无
```

输出格式：

```
若干行，每行为一个可能解，依次输出鸡的数目、鸡的脚数、兔的数目、兔的脚数
```

代码：

```c
#include <stdio.h>
#define true 1
#define false 0
typedef char bool;

int chick_feet[720+1], rabbit_feet[720+1], tmp_perm[6];
bool visit[6];
int cnt, rabbit_cnt, chick_cnt;

void all_permutation(int depth) {
    if (depth == 6) {
        int sum_chick = 0, sum_rabbit = 0;
        for (int i = 0; i < 3; i++) {
            sum_chick *= 10; sum_rabbit *= 10;
            sum_chick += tmp_perm[i]; sum_rabbit += tmp_perm[i+3];
        }
        chick_feet[++cnt] = sum_chick; rabbit_feet[cnt] = sum_rabbit;
        return;
    }
    for (int i = 0; i < 6; i++) {
        if (visit[i]) continue;
        visit[i] = 1;
        tmp_perm[depth] = i;
        all_permutation(depth+1);
```

```
        visit[i] = 0;
    }
    return;
}

int main() {
    all_permutation(0);
    for (int i=1; i<=cnt; i++) {
        if (!(chick_feet[i] / 100) || !(rabbit_feet[i] / 100)) continue;
        if (chick_feet[i] % 2 || rabbit_feet[i] % 4) continue;
        chick_cnt = chick_feet[i] / 2; rabbit_cnt = rabbit_feet[i] / 4;
        if (chick_cnt == rabbit_cnt) {
            printf("Chick: %d, Chick_feet: %d, Rabbit: %d, Rabbit_feet: %d\n", \
                chick_cnt, chick_feet[i], rabbit_cnt, rabbit_feet[i]);
        }
    }
    return 0;
}
```

输入1：

```

```

输出1：

```
Chick: 76, Chick_feet: 152, Rabbit: 76, Rabbit_feet: 304
```

Optional-Experiment8-2

题目：

```
编程计算 n=50 时 S 与 NS 的值，并比较大小：如果 S>NS，输出 1；S=NS，输出 0；S<NS，输出
-1。
```

输入格式：

```
无
```

输出格式：

```
一行，一个整数，为 -1，0 或 1
```

代码：

```
#include <stdio.h>

double calc_S(int n) {
    int S = 0, sub_S = 0;
    for(int k=1; k<=n; ++k) {
        sub_S += k * k;
        S += sub_S * k;
    }
```

```
    return S;
}

double calc_NS(int n) {
    return (double)(n) * (n+1) * (n+2) * (8*n*n + 11*n + 1) / 120;
}

int main() {
    int n = 50;
    double S = calc_S(n), NS = calc_NS(n);
    if (S>NS) puts("1");
    else if (S<NS) puts("-1");
    else puts("0");
    return 0;
}
```

输入1：

```

```

输出1：

```
0
```