Experiment9-董皓彧

环境：

```
gcc.exe (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
Visual Stdio Code 1.84.2
```

作业仓库地址：
https://github.com/FHYQ-Dong/Tsinghua-Program-Design-Assignments/tree/main/Experiment9

必做题

Experiment9-1

题目：

> 设有若干只羊（>100 只），需要从中挑选最肥的 100 只供使用。如何选拔这 100 只最肥的羊，请同学帮忙使用至少 2 种不同算法编程。提示：羊的重量可以采用随机函数 rand() 来产生。另外：使用断点调试，截获保存“最大 100 只羊”数组的调试窗口图，并在作业报告中提交。

输入格式：

> 共 2 行，第一行为羊的总数，第二行为每只羊的重量，用空格隔开

输出格式：

> 分 Answer1 和 Answer2 两部分，具体见输出

代码：

```c
#include <stdio.h>
#define INF 0x7fffffff

typedef struct Sheep {
    int weight, idx;
} Sheep;

void swap_sheep(Sheep *a, Sheep *b) {
    Sheep tmp = *a; *a = *b; *b = tmp;
}
int max(int a, int b) {
    return a > b ? a : b;
}


void qsort_sheep(Sheep *beg, Sheep *end) { // big-endian
    if (beg >= end) return;
    Sheep *l = beg, *r = end - 1, *p = beg;
    while (l < r) {
```

```
        while (l < r && r->weight <= p->weight) r--;
        while (l < r && l->weight >= p->weight) l++;
        if (l < r) swap_sheep(l, r);
    }
    swap_sheep(p, l);
    qsort_sheep(beg, l);
    qsort_sheep(l + 1, end);
}

// Solution 1
Sheep* find_loop(Sheep s[], int n) {
    static Sheep mxSheep[102] = {-1, 0};
    mxSheep[101].weight = INF;
    for (int i=1; i<=n; ++i) {
        for (int j=1; j<=100; ++j) {
            if (mxSheep[j].weight <= s[i].weight) {
                mxSheep[j-1] = mxSheep[j];
                if (mxSheep[j+1].weight > s[i].weight) {
                    mxSheep[j] = s[i];
                    break;
                }
            }
        }
    }
    return mxSheep;
}

// Solution 2
Sheep* find_sort(Sheep s[], int n) {
    qsort_sheep(s+1, s+n+1);
    return s;
}

int main() {
    int n;
    Sheep s[(int)1e5+5];
    scanf("%d", &n);
    for(int i=1; i<=n; ++i) scanf("%d", &s[i].weight), s[i].idx = i;
    if (n < 100) {
        printf("Error: n < 100\n");
        return 0;
    }

    // Solution 1
    Sheep* mxSheep = find_loop(s, n);
    printf("Answer1 :\nWeight\tIndex\n----------------\n");
    for (int i=100; i>=1; --i) printf("%d\t%d\n", mxSheep[i].weight,
mxSheep[i].idx);

    // blank line
    printf("\n");

    // Solution 2
    mxSheep = find_sort(s, n);
    printf("Answer2 :\nWeight\tIndex\n----------------\n");
```

```
    for (int i=1; i<=100; ++i) printf("%d\t%d\n", mxSheep[i].weight,
mxSheep[i].idx);

    return 0;
}
```

输入1：

```
290
6870 3247 421 9743 191 3141 5793 8262 1183 5789 6887 4617 7586 6574 1953 911 8757
2980 6128 3394 9209 1902 6403 2566 7716 2135 7147 4090 6121 1255 2972 1069 6377
6618 3191 4611 2106 7378 1590 1528 8743 9385 8998 5028 7480 4521 5148 7247 8771
1679 2370 7782 4451 8349 4705 7071 1912 9231 2371 4314 9119 561 73 7273 8644 1919
4634 1761 1189 5917 7677 4321 4200 5384 2217 8851 2464 589 1035 497 9228 584 9097
5881 5825 2911 645 4852 8777 2976 915 2643 2539 6088 8814 7643 2845 8118 3271
9363 2200 2672 512 1627 6331 7348 6824 6207 6531 993 7464 6380 7769 1575 8238
1795 1979 6764 583 2252 1936 7921 5619 9820 3414 469 2259 5642 9551 6168 9974
1918 5980 4386 3589 1147 3607 5284 2683 159 6983 2015 8249 569 3424 4677 5145
4266 2106 6685 2385 3871 1940 5066 8906 2926 6551 862 514 222 1791 3467 3503 9644
1144 634 9141 2684 1429 3952 7545 8413 1671 5534 5943 8063 4923 1896 9217 3913
9781 2280 9881 3638 1805 4627 8322 7520 2839 3868 6881 9665 6759 1096 4969 4629
6852 8290 8136 8756 4892 9024 1516 542 4520 3494 7370 4887 690 8921 512 4294 8052
1526 8409 969 8707 3268 2065 5499 6476 2312 6170 2201 5806 7656 606 833 8344 8129
7509 2907 3252 8156 5354 3532 4468 6207 8089 2553 3308 8961 6972 8074 6048 6928
7575 6303 1215 6870 5152 8935 2208 8456 9155 8158 2748 9889 1764 5158 1405 5455
1238 2474 2693 5453 6636 1775 9298 1281 9630 1900 2289 3366 3358 8092 8976 2124
188 1105 3128 527 3255 4539 2555 3558 884 585 8330 2060
```

输出1：

```
Answer1 :
Weight  Index
----------------
9974    131
9889    258
9881    183
9820    124
9781    181
9743    4
9665    192
9644    164
9630    271
9551    129
9385    42
9363    100
9298    269
9231    58
9228    81
9217    179
9209    21
9155    255
9141    167
9119    61
9097    83
```

9024 202
8998 43
8976 277
8961 242
8935 252
8921 210
8906 155
8851 76
8814 95
8777 89
8771 49
8757 17
8756 200
8743 41
8707 217
8644 65
8456 254
8413 172
8409 215
8349 54
8344 229
8330 289
8322 187
8290 198
8262 8
8249 143
8238 115
8158 256
8156 234
8136 199
8129 230
8118 98
8092 276
8089 239
8074 244
8063 176
8052 213
7921 122
7782 52
7769 113
7716 25
7677 71
7656 226
7643 96
7586 13
7575 247
7545 171
7520 188
7509 231
7480 45
7464 111
7378 38
7370 207
7348 106
7273 64

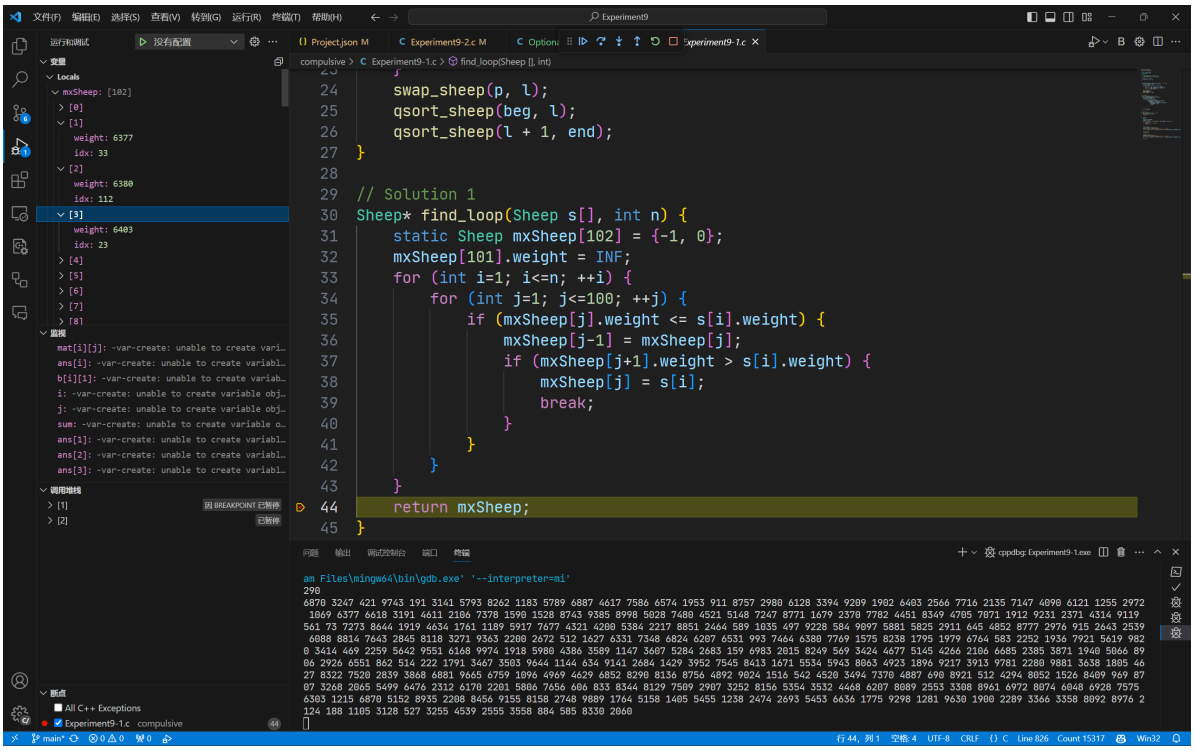
```
7247    48
7147    27
7071    56
6983    141
6972    243
6928    246
6887    11
6881    191
6870    250
6870    1
6852    197
6824    107
6764    118
6759    193
6685    150
6636    267
6618    34
6574    14
6551    157
6531    109
6476    221
6403    23
6380    112
6377    33

Answer2 :
Weight  Index
----------------
9974    131
9889    258
9881    183
9820    124
9781    181
9743    4
9665    192
9644    164
9630    271
9551    129
9385    42
9363    100
9298    269
9231    58
9228    81
9217    179
9209    21
9155    255
9141    167
9119    61
9097    83
9024    202
8998    43
8976    277
8961    242
8935    252
8921    210
```

8906 155
8851 76
8814 95
8777 89
8771 49
8757 17
8756 200
8743 41
8707 217
8644 65
8456 254
8413 172
8409 215
8349 54
8344 229
8330 289
8322 187
8290 198
8262 8
8249 143
8238 115
8158 256
8156 234
8136 199
8129 230
8118 98
8092 276
8089 239
8074 244
8063 176
8052 213
7921 122
7782 52
7769 113
7716 25
7677 71
7656 226
7643 96
7586 13
7575 247
7545 171
7520 188
7509 231
7480 45
7464 111
7378 38
7370 207
7348 106
7273 64
7247 48
7147 27
7071 56
6983 141
6972 243
6928 246

```
6887    11
6881    191
6870    1
6870    250
6852    197
6824    107
6764    118
6759    193
6685    150
6636    267
6618    34
6574    14
6551    157
6531    109
6476    221
6403    23
6380    112
6377    33
```

调试界面：



Experiment9-2

题目：

```
Gauss 消元
```

输入格式：

```
若干行：
第一行：一个正整数 n，表示未知数和方程的数量；
接下来 n 行：每行 n+1 个数，表示方程组的增广矩阵
```

输出格式：

```
共 2n+4 行:
前 n+1 行输出方程组的系数矩阵;
第 n+2 行、n+3 行输出方程组的常数项;
第 n+4 行至第 2n+4 行输出方程组的解
```

代码:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define NO_PIVOT -1 // in func select_pivot
#define INF_ANS 1
#define NO_ANS -1
#define SINGLE_ANS 0
typedef int ANS_TYPE;


void swap_line(double** a, double** b) { double* tmp = *a; *a = *b; *b = tmp;
return; }
double dabs(double x) { return x>0 ? x : -x; }

int select_pivot(double** mat, double** b, int line_cnt, int cur_column) { // if
no pivot, return -1

    double mx_abs = 0;
    int mx_row = cur_column;

    for (int i=cur_column; i<=line_cnt; ++i) {
        if (dabs(mat[i][cur_column]) > mx_abs) {
            mx_abs = dabs(mat[i][cur_column]);
            mx_row = i;
        }
    }

    if (mx_abs == 0) return NO_PIVOT;

    swap_line(&mat[cur_column], &mat[mx_row]);
    swap_line(&b[cur_column], &b[mx_row]);
    return cur_column;
}

void normalize_line(double* line, double pivot, int pivot_pos, int maxpos) {
    for (int i=pivot_pos; i<=maxpos; ++i) line[i] /= pivot;
    return;
}

void eliminate_line(double* target_line, double* pivot_line, int times, int
maxpos) {
    for (int i=1; i<=maxpos; ++i) target_line[i] -= times * pivot_line[i];
    return;
}

ANS_TYPE get_ans(double** mat, double** b, int mat_size, double* ans) {
```

```
    // Infinite answers
    if (b[mat_size][1]==0 && mat[mat_size][mat_size]==0) return INF_ANS;

    // No answer
    if (b[mat_size][1]!=0 && mat[mat_size][mat_size]==0) return NO_ANS;

    // Single answer
    ans[mat_size] = b[mat_size][1] / mat[mat_size][mat_size];
    for (int i=mat_size-1; i>=1; --i) {
        double sum = 0;
        for (int j=i+1; j<=mat_size; ++j) sum += mat[i][j] * ans[j];
        ans[i] = (b[i][1] - sum);
    }
    return SINGLE_ANS;
}

void print_mat_a(double** mat, int mat_size) {
    printf("Mat A = \n");
    for (int i=1; i<=mat_size; ++i) {
        for (int j=1; j<=mat_size; ++j) printf("\t%.6lf", mat[i][j]);
        printf("\n");
    }
    return;
}
void print_mat_b(double** b, int mat_size) {
    printf("Mat B = \n");
    for (int i=1; i<=mat_size; ++i) printf("\t%.6lf", b[i][1]);
    printf("\n");
    return;
}

int main() {

    // init
    int mat_size;
    scanf("%d", &mat_size);
    double** mat = (double**)malloc(sizeof(double) * (mat_size+1));
    for (int i=0; i<=mat_size; ++i) mat[i] = (double*)malloc(sizeof(double) *
(mat_size+1));
    for (int i=0; i<=mat_size; ++i) memset(mat[i], 0, sizeof(double) *
(mat_size+1));
    double** b = (double**)malloc(sizeof(double) * (mat_size+1));
    for (int i=0; i<=mat_size; ++i) b[i] = (double*)malloc(sizeof(double) * 2);
    for (int i=0; i<=mat_size; ++i) memset(b[i], 0, sizeof(double) * 2);

    // input MAT_A and MAT_B
    for (int i=1; i<=mat_size; ++i) {
        for (int j=1; j<=mat_size; ++j) scanf("%lf", &mat[i][j]);
        scanf("%lf", &b[i][1]);
    }

    // print MAT_A and MAT_B
    print_mat_a(mat, mat_size);
    print_mat_b(b, mat_size);
```

```
    // Gauss Elimination
    for (int column_i=1; column_i<=mat_size; ++column_i) { // column i

        // select pivot
        int pivot_line = select_pivot(mat, b, mat_size, column_i);
        if (pivot_line == NO_PIVOT) { // infinite answers
            printf("Infinite answers.\n");
            goto FREE;
        }

        // normalize pivot line
        double pivot = mat[pivot_line][column_i];
        normalize_line(mat[pivot_line], pivot, column_i, mat_size); // normalize
pivot line (MAT A)
        normalize_line(b[pivot_line], pivot, 1, 1); // normalize pivot line (MAT
B)

        // eliminate other lines
        for (int line_j=pivot_line+1; line_j<=mat_size; ++line_j) {
            double times = mat[line_j][column_i];
            eliminate_line(mat[line_j], mat[pivot_line], times, mat_size);
            eliminate_line(b[line_j], b[pivot_line], times, 1);
        }
    }

    // get answer
    double* ans = (double*)malloc(sizeof(double) * (mat_size+1));
    memset(ans, 0, sizeof(double) * (mat_size+1));

    ANS_TYPE ans_type = get_ans(mat, b, mat_size, ans);
    if (ans_type == INF_ANS) printf("Infinite answers.\n");
    else if (ans_type == NO_ANS) printf("No answer.\n");
    else {
        printf("Answer:\n");
        for (int i=1; i<=mat_size; ++i) printf("x(%d) = %.6lf\n", i, ans[i]);
        printf("\n");
    }

    // free
    FREE:
    for (int i=0; i<=mat_size; ++i) free(mat[i]), free(b[i]);
    free(mat), free(b), free(ans);
    return 0;
}
```

输入1：

```
3
1 2 3 1
3 4 5 2
3 4 6 3
```

输出1：

```
Mat A =
    1.000000    2.000000    3.000000
    3.000000    4.000000    5.000000
    3.000000    4.000000    6.000000
Mat B =
    1.000000    2.000000    3.000000
Answer:
x(1) = 1.000000
x(2) = -1.500000
x(3) = 1.000000
```

Experiment9-3

题目：

```
小朋友解鸡兔同笼
```

输入格式：

```
共 1 行：两个正整数 m 和 n，分别表示鸡和兔的总数、脚的总数
```

输出格式：

```
共 1 行：兔子的数目、鸡的数目
```

代码：

```c
#include <stdio.h>

int main() {
    int animal_cnt = 0, feet_cnt = 0;
    scanf("%d%d", &animal_cnt, &feet_cnt);
    int rabbit_cnt = 0, chicken_cnt = 0;

    if (feet_cnt % 2 == 1) {
        printf("No answer.\n");
        return 0;
    }
    rabbit_cnt = (feet_cnt - 2 * animal_cnt) / 2; // 抬起两只脚
    chicken_cnt = animal_cnt - rabbit_cnt;
    if (rabbit_cnt < 0 || chicken_cnt < 0) {
        printf("No answer.\n");
        return 0;
    }
    printf("Rabbit: %d\nChicken: %d\n", rabbit_cnt, chicken_cnt);
    return 0;
}
```

输入1：

```
20 60
```

输出1：

```
Rabbit: 10
Chicken: 10
```

选做题

Optional-Experiment9-1

题目：

> 满足如下条件的正整数称为“幸福数”：计算正整数各位数字的平方和，如果计算结果不为 1，则对该结果进行类似的计算直到结果为 1。如果计算过程中出现循环并且不包含 1，则原来的正整数不是幸福数。
> 要求：输入一个正整数，如果是幸福数则输出 1，否则输出 0

输入格式：

> 一行，一个正整数

输出格式：

> 一行，一个整数，为 0 或 1

代码：

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

typedef struct Node {
    int val;
    struct Node *left, *right;
} Node;
typedef struct BST {
    Node *root;
    Node* (*find)(int val, struct BST* bst);
    void (*insert)(int val, struct BST* bst);
    void (*free)(struct BST* bst);
} BST;

Node *new_Node(int val);
BST* new_BST();
Node* _find(int val, BST *bst);
void _insert(int val, BST *bst);
void _free_bst(BST *bst);
void _free_node(Node *node);
int split(int a);

int main() {
    int a; scanf("%d", &a);
    BST *bst = new_BST(new_Node(a));
    while(a >= 10) {
        a = split(a);
```

```
        if (bst->find(a, bst)) {
            printf("0\n");
            return 0;
        }
        bst->insert(a, bst);
    }
    if (a == 1) printf("1\n");
    else printf("0\n");

    bst->free(bst);
    return 0;
}

int split(int a) {
    int ans = 0;
    while (a) {
        ans += (a % 10) * (a % 10);
        a /= 10;
    }
    return ans;
}

Node* new_Node(int val) {
    Node *node = (Node*)malloc(sizeof(Node));
    node->val = val;
    node->left = NULL; node->right = NULL;
    return node;
}

BST* new_BST(Node* root) {
    BST *bst = (BST*)malloc(sizeof(BST));
    bst->root = root;
    bst->find = _find;
    bst->insert = _insert;
    bst->free = _free_bst;
    return bst;
}

Node* _find(int val, BST *bst) {
    Node *cur = bst->root;
    while (cur) {
        if (cur->val == val) return cur;
        else if (cur->val > val) cur = cur->left;
        else cur = cur->right;
    }
    return NULL;
}

void _insert(int val, BST *bst) {
    Node *cur = bst->root;
    while (cur) {
        if (cur->val == val) return;
        else if (cur->val > val) {
            if (cur->left) cur = cur->left;
            else {
```

```
                cur->left = new_Node(val);
                return;
            }
        }
        else {
            if (cur->right) cur = cur->right;
            else {
                cur->right = new_Node(val);
                return;
            }
        }
    }
    return;
}

void _free_bst(BST *bst) {
    _free_node(bst->root);
    free(bst);
    return;
}

void _free_node(Node *node) {
    if (!node) return;
    _free_node(node->left);
    _free_node(node->right);
    free(node);
    return;
}
```

输入1：

```
13
```

输出1：

```
1
```

输入2：

```
12313
```

输出2：

```
0
```

输入3：

```
9534032
```

输出3：

```
0
```

输入4：

```
2342
```

输出4：

```
0
```

输入5：

```
1
```

输出5：

```
1
```

Optional-Experiment9-2

题目：

```
执行下列程序的输出结果是_____。
```

输入格式：

```
无
```

输出格式：

```
输出结果
```

代码：

```
#include <stdio.h>

int main() {
    printf("10,9,8,7,6,1,2,3,4,5\n");
    return 0;
}
```

输入1：

```

```

输出1：

```
10,9,8,7,6,1,2,3,4,5
```

Optional-Experiment9-3

题目：

> 编写程序，输入一个 18 位的身份证号，输出该身份证号是否合法。如果非法，输出正确的校验位

输入格式：

> 一行，一个 18 位的身份证号

输出格式：

```
共 1 或 2 行：
若合法，输出 1 行：Right
若非法，输出 2 行：
    第 1 行：Wrong
    第 2 行：正确的校验位
```

代码：

```c
#include <stdio.h>

int main() {
    char id_str[19]; scanf("%s", id_str);
    int id[18], weight[18] = {-1,7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2}, sum = 0;

    for (int i=0; i<=16; ++i) id[i+1] = id_str[i] - '0';
    for (int i=1; i<=17; ++i) {
        sum += (id[i] * weight[i]) % 11;
        sum %= 11;
    }

    int chk = (12 - sum) % 11;
    if (chk == 10) {
        if (id_str[17] == 'X') printf("Right\n");
        else printf("Wrong\nCorrect: X\n");
    }
    else {
        if (id_str[17] == chk + '0') printf("Right\n");
        else printf("Wrong\nCorrect: %d\n", chk);
    }
    return 0;
}
```

输入1：

```
111111111111111111
```

输出1：

```
Wrong
Correct: 0
```

输入2:

```
110108200001019999
```

输出2:

```
Wrong
Correct: 2
```

输入3:

```
110108200001019992
```

输出3:

```
Right
```

输入4:

```
330326199211027412
```

输出4:

```
Right
```