第 3 次编程练习报告

一、编程练习 1——中国剩余定理

➢ 源码部分：

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

void extended_euclidean_algorithm(int a, int b, int* gcd, int* x, int* y) {
    // a和b的最大公约数
    int r, q, x1, x2, y1, y2;
    if (b == 0) {
        *gcd = a;
        *x = 1;
        *y = 0;
        return;
    }
    x2 = 1; x1 = 0; y2 = 0; y1 = 1;
    while (b > 0) {
        q = a / b;
        r = a - q * b;
        *x = x2 - q * x1;
        *y = y2 - q * y1;
        a = b;
        b = r;
        x2 = x1;
        x1 = *x;
        y2 = y1;
        y1 = *y;
    }
    *gcd = a;
    *x = x2;
    *y = y2;
}

int chinese_remainder_theorem(int* a, int* m, int n) {
    int M = 1, x = 0, gcd, y;
    for (int i = 0; i < n; i++) {
        M *= m[i];
```

```
    }
    for (int i = 0; i < n; i++) {
        int Mi = M / m[i];
        extended_euclidean_algorithm(Mi, m[i], &gcd, &y, &y);
        x += a[i] * Mi * y;
    }
    return x % M;
}

int main() {
    int n;
    printf("请输入同余方程个数n：");
    scanf_s("%d", &n);
    int* a = (int*)malloc(n * sizeof(int));
    int* m = (int*)malloc(n * sizeof(int));
    printf("请输入同余方程：\n");
    for (int i = 0; i < n; i++) {
        printf("x ≡ ");
        scanf_s("%d", &a[i]);
        printf("(mod ");
        scanf_s("%d", &m[i]);
        printf(")\n");
    }
    int x = chinese_remainder_theorem(a, m, n);
    printf("x ≡ %d (mod ", x);
    for (int i = 0; i < n; i++) {
        printf("%d", m[i]);
        if (i != n - 1) {
            printf(",");
        }
    }
    printf(")\n");
    free(a);
    free(m);
    system("pause");
    return 0;
}
```

➢ 说明部分：

这段代码实现了中国剩余定理。

1.kr 和 kq 为全局变量，分别记录循环中商和余数数组的下标，初始值为 2 和 1。它们在循环中用于记录每次欧几里得算法迭代的商和余数。

2.ojld 函数实现了欧几里得算法，用于计算两个数的最大公约数。参数 a 和 b 为需要计算最大公约数的两个数，res 数组用于记录每次欧几里得算法迭代的余数，q 数组用于存储

每次迭代的商，s 和 t 数组用于存储最终得到的贝祖等式中的系数。函数中，首先将 s 和 t 的初值设置为 1 和 0 以及 0 和 1，分别对应于上一次和这一次迭代的结果，然后将 a 和 b 取绝对值并分别存储在 max 和 min 变量中。接着，在 while 循环中进行欧几里得算法迭代计算，直至余数为 0。在每次迭代中，计算商和余数，并将商存储在 q 数组中，将余数存储在 res 数组中，然后根据贝祖等式迭代计算 s 和 t，并将它们存储在相应的数组中。最后，更新 max 和 min 的值，将 max 设置为原来的 min，将 min 设置为余数。在每次迭代结束后，kr 和 kq 分别自增 1，以便记录下一次迭代的商和余数。

3.在 main 函数中，首先输入需要计算的方程组的个数 n，以及每个方程的 b 和 m 的值，分别存储在动态分配的 b 和 m 数组中。然后，根据所有方程的 m 值计算出整个方程组的 M 值，存储在动态分配的 M 数组中。接着，动态分配所需的数组空间，并定义一个变量 x，用于存储最终结果。在接下来的循环中，调用 ojld 函数计算出每个方程中的 s 和 t 值，并根据中国剩余定理的公式，计算出方程组的解。循环结束后，输出结果并释放动态分配的数组空间。

4.在输出结果前，需要对 s 和 t 的值进行一些调整。如果 s 或 t 为负数，则需要加上一个 b 或 m 的值，以保证它们为正数。

➢ 运行示例：