

Experiment6-董皓彧

环境：

```
gcc.exe (x86_64-win32-seh-rev0, Built by MinGW-w64 project) 8.1.0  
Visual Studio Code 1.83.1
```

作业仓库地址：

<https://github.com/FHYQ-Dong/Tsinghua-Program-Design-Assignments/tree/main/Experiment6>

必做题

Experiment6-1

题目：

求100以内的孪生素数

输入格式：

无

输出格式：

若干行，每行两个数，为孪生素数，小数在前

代码：

```
#include<stdio.h>  
#define true 1  
#define false 0  
typedef int bool;  
  
bool is_prime[105];  
  
void ertosthenes() {  
    for (int i = 2; i <= 100; i++) {  
        is_prime[i] = true;  
    }  
    for (int i = 2; i <= 100; i++) {  
        if (is_prime[i]) {  
            for (int j = i * 2; j <= 100; j += i) {  
                is_prime[j] = false;  
            }  
        }  
    }  
    return;  
}
```

```
int main() {
    ertosthenes();
    for(int i=2; i<=100; ++i) {
        if(is_prime[i] && is_prime[i+2]) {
            printf("%d %d\n", i, i+2);
        }
    }
    return 0;
}
```

输入1:

输出1:

```
3 5
5 7
11 13
17 19
29 31
41 43
59 61
71 73
```

Experiment6-2

题目:

给出一个菱形的对角线长 l ，打印这个菱形。若 l 不是大于2的奇数，输出Input Error

输入格式:

一个数, l

输出格式:

菱形或Input Error

代码:

```
#include<stdio.h>

int main() {
    int x;
    scanf("%d", &x);
    if (x % 2 == 0 || x <= 2) printf("Input Error");
    else {
        for (int i=1; i<=(x+1)/2; ++i) {
            for (int j=1; j<=(x - 2*i + 1) / 2; ++j) printf(" ");
            for (int j=1; j<=i*2-1; ++j) printf("*");
            printf("\n");
        }
    }
}
```

```

    }
    for(int i=(x+1)/2-1; i>=1; --i) {
        for (int j=1; j<=(x - 2*i + 1) / 2; ++j) printf(" ");
        for (int j=1; j<=i*2-1; ++j) printf("*");
        printf("\n");
    }
}
return 0;
}

```

输入1:

1

输出1:

Input Error

输入2:

3

输出2:

```

*
***
*

```

输入3:

7

输出3:

```

*
***
*****
*****
*****
***
*

```

输入4:

10

输出4:

Input Error

输入5:

输出5:

```
*  
  
***  
  
*****  
  
*******  
  
*********  
  
*****  
  
***  
  
*
```

Experiment6-3

题目:

验证哥德巴赫猜想：给出一个不小于4的偶数N，输出两个质数a、b，使得 $N=a+b$ 。若N不是不小于4的偶数，输出Input Error

输入格式:

一行，一个正整数

输出格式:

一行，格式为 $N=a+b$

代码:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define true 1
#define false 0
typedef int bool;

bool* ertosthenes(bool *is_prime, int N) {
    for (int i = 2; i <= N; i++) {
        if (is_prime[i]) {
            for (int j = i * 2; j <= N; j += i) {
                is_prime[j] = false;
            }
        }
    }
    return is_prime;
}
```

```

int main() {
    int N;
    scanf("%d", &N);

    if (N % 2 == 1 || N < 4) {
        printf("Input Error");
        return 0;
    }

    int* is_prime = (int*)malloc(sizeof(int) * (N + 1));

    for (int i = 0; i <= N; ++i) is_prime[i] = true;
    is_prime = ertosthenes(is_prime, N);

    for (int i = 2; i <= N; ++i) {
        if(is_prime[i] && is_prime[N-i]) {
            printf("%d=%d+%d\n", N, i, N-i);
            return 0;
        }
    }
    return 0;
}

```

输入1:

2

输出1:

Input Error

输入2:

4

输出2:

4=2+2

输入3:

11

输出3:

Input Error

输入4:

10032

输出4:

10032=23+10009

输入5:

998244

输出5:

998244=7+998237

输入6:

165432

输出6:

165432=41+165391

输入7:

8465134

输出7:

8465134=3+8465131

输入8:

5646872

输出8:

5646872=3+5646869

输入9:

8674532

输出9:

8674532=43+8674489

选做题

Optional-Experiment6-1

题目：

某幼儿园按如下方法依次给A、B、C、D、E五个小孩发苹果。将全部苹果的一半再加二分之一一个苹果发给第一个小孩；将剩下苹果的三分之一再加三分之一一个苹果发给第二个小孩；将剩下苹果的四分之一再加四分之一一个苹果发给第三个小孩；将剩下苹果的五分之一再加五分之一一个苹果发给第四个小孩；将最后剩下的11个苹果发给第五个小孩。每个小孩得到的苹果数均为整数。确定原来共有多少个苹果？每个小孩各得到多少个苹果？

输入格式：

无

输出格式：

6行，分别为：原来共有多少个苹果、每个小孩各得到多少个苹果

代码：

```
#include<stdio.h>

int get_all_apples(int cur_child) { // 递归地求分给第 i 个孩子之前还剩下多少苹果
    if (cur_child == 5) return 11;
    else return (1 + (cur_child + 1) * get_all_apples(cur_child + 1)) /
cur_child; // 通过手算得到递归式
}

void split_apples(int cur_apples) { // 分苹果
    for (int i=1; i<=4; ++i) {
        int nth_child_apples = (cur_apples + 1) / (i+1); // 第 i 个孩子分到的苹果数
        printf("The %dth child gets %d apples.\n", i, nth_child_apples);
        cur_apples -= nth_child_apples;
    }
    printf("The last child gets %d apples.\n", cur_apples); // 最后一个孩子分到的苹果数需要单独处理
    return;
}

int main() {
    int all_apples = get_all_apples(1);
    printf("The total number of apples is %d.\n", all_apples);
    split_apples(all_apples);
    return 0;
}
```

输入1:

输出1:

The total number of apples is 59.
The 1th child gets 30 apples.
The 2th child gets 10 apples.
The 3th child gets 5 apples.
The 4th child gets 3 apples.
The last child gets 11 apples.

Optional-Experiment6-2

题目：

$k(i) = i!$, $s(i) = k(1) + k(2) + \dots + k(i)$, 输入正整数 n ($1 \leq n \leq 10$) , 输出 $s(n)$ 的值

输入格式：

一个正整数 n ($1 \leq n \leq 10$)

输出格式：

一行, $s(n) = \text{ans}$

代码：

```
#include<stdio.h>

int main() {
    int s[11] = {0}, k[11] = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    int n;
    scanf("%d", &n);
    for (int i=1; i<=n; ++i) {
        k[i] = k[i-1] * i;
        s[i] = s[i-1] + k[i];
    }
    printf("s(%d) = %d\n", n, s[n]);
    return 0;
}
```

输入1：

1

输出1：

$s(1) = 1$

输入2：

2

输出2：

$$s(2) = 3$$

输入3:

3

输出3:

$$s(3) = 9$$

输入4:

4

输出4:

$$s(4) = 33$$

输入5:

5

输出5:

$$s(5) = 153$$

输入6:

6

输出6:

$$s(6) = 873$$

输入7:

7

输出7:

$$s(7) = 5913$$

输入8:

8

输出8:

$s(8) = 46233$

输入9:

9

输出9:

$s(9) = 409113$

输入10:

10

输出10:

$s(10) = 4037913$

Optional-Experiment6-3

题目:

找出1-1000中仅包含5个因子（包括1和自身）的所有自然数，输出这些自然数的所有因子

输入格式:

无

输出格式:

若干行，每行第一个数为符合要求的自然数，之后为该自然数的所有因子

代码:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node Node;
typedef struct Num Num;

struct Node {
    int val;
    Node* next;
};

struct Num {
    int cnt;
    Node* head;
};

Num num[1001];
```

```

int main() {
    for (int i=1; i<=1000; ++i) {
        for (int j=1; i*j<=1000; ++j) {
            if (num[i*j].cnt > 5) continue;
            num[i*j].cnt++;
            Node* tmp = (Node*)malloc(sizeof(Node));
            tmp->val = i;
            tmp->next = num[i*j].head;
            num[i*j].head = tmp;
        }
    }
    printf("自然数\t因数\n");
    for (int i=1; i<=1000; ++i) {
        if (num[i].cnt == 5) {
            printf("%d\t", i);
            Node* tmp = num[i].head;
            while (tmp) {
                printf("%d ", tmp->val);
                tmp = tmp->next;
            }
            printf("\n");
        }
    }
    return 0;
}

```

输入1:

输出1:

```

自然数 因数
16  16 8 4 2 1
81  81 27 9 3 1
625 625 125 25 5 1

```

Optional-Experiment6-4

题目:

有A, B, C, D, E, F 六个小朋友, 现将三顶相同的白帽子, 三顶相同的黑帽子分给他们, 每人一顶。请编写程序计算不同分配方案的个数, 并打印所有的分配方案

输入格式:

无

输出格式:

若干行, 第一行为总方案数, 之后每一行为一种方案

代码：

```
#include<stdio.h>

int main() {
    int black[4] = {0}, white[4] = {0};
    char children[7] = {'0', 'A', 'B', 'C', 'D', 'E', 'F'};
    int cnt_all = 0;
    printf("总方案数: %d\n", 20);
    for (int i=1; i<=6; ++i) {
        for (int j=i+1; j<=6; ++j) {
            for(int k=j+1; k<=6; ++k) {
                black[1] = i; black[2] = j; black[3] = k;
                int cnt_white = 0;
                for (int m=1; m<=6; ++m) {
                    if (m == i || m == j || m == k) continue;
                    white[++cnt_white] = m;
                }
                printf("第%d种方案: 黑帽子: %c%c%c, 白帽子: %c%c%c\n", ++cnt_all,
                    children[black[1]], children[black[2]], children[black[3]], children[white[1]],
                    children[white[2]], children[white[3]]);
            }
        }
    }
    return 0;
}
```

输入1:

输出1:

```
总方案数: 20
第1种方案: 黑帽子: ABC, 白帽子: DEF
第2种方案: 黑帽子: ABD, 白帽子: CEF
第3种方案: 黑帽子: ABE, 白帽子: CDF
第4种方案: 黑帽子: ABF, 白帽子: CDE
第5种方案: 黑帽子: ACD, 白帽子: BEF
第6种方案: 黑帽子: ACE, 白帽子: BDF
第7种方案: 黑帽子: ACF, 白帽子: BDE
第8种方案: 黑帽子: ADE, 白帽子: BCF
第9种方案: 黑帽子: ADF, 白帽子: BCE
第10种方案: 黑帽子: AEF, 白帽子: BCD
第11种方案: 黑帽子: BCD, 白帽子: AEF
第12种方案: 黑帽子: BCE, 白帽子: ADF
第13种方案: 黑帽子: BCF, 白帽子: ADE
第14种方案: 黑帽子: BDE, 白帽子: ACF
第15种方案: 黑帽子: BDF, 白帽子: ACE
第16种方案: 黑帽子: BEF, 白帽子: ACD
第17种方案: 黑帽子: CDE, 白帽子: ABF
第18种方案: 黑帽子: CDF, 白帽子: ABE
第19种方案: 黑帽子: CEF, 白帽子: ABD
第20种方案: 黑帽子: DEF, 白帽子: ABC
```

Optional-Experiment6-5

题目：

从键盘输入一个五位正整数，首先分离出该正整数中的每一位数字，然后用分离出的每位数字组成一个最接近40000的数和一个最接近60000的数。要求检查输入数据的合法性

输入格式：

一个正整数

输出格式：

一行或两行。若输入数据不合法，输出Input Error；若输入数据合法，输出两行，分别为最接近40000的数和最接近60000的数

代码：

```
#include<stdio.h>

int x, cnt, ans4, ans6;
int digit[6], alternative[121];

int abs(int x) {
    return x > 0 ? x : -x;
}
int min(int x, int y) {
    return x < y ? x : y;
}

void all_permutation(int ans, int depth) {
    if (depth == 5) {
        alternative[++cnt] = ans;
        return;
    }
    for(int i=1; i<=5; ++i) {
        if (digit[i] == -1) continue;
        int tmp = digit[i];
        digit[i] = -1;
        all_permutation(ans*10+tmp, depth+1);
        digit[i] = tmp;
    }
    return;
}

int main() {
    scanf("%d", &x);
    ans4 = x; ans6 = x;
    if (x <= 9999 || x >= 100000) {
        printf("Input Error\n");
        return 0;
    }
    for (int i=1; i<=5; ++i) {
```

```

        digit[i] = x % 10;
        x /= 10;
    }
    all_permutation(0, 0);
    for (int i=1; i<=cnt; ++i) {
        if (abs(ans4 - 40000) > abs(alternative[i] - 40000)) ans4 =
alternative[i];
        if (abs(ans6 - 60000) > abs(alternative[i] - 60000)) ans6 =
alternative[i];
    }
    printf("ans4 = %d\n", ans4);
    printf("ans6 = %d\n", ans6);
    return 0;
}

```

输入1:

1

输出1:

Input Error

输入2:

1000000

输出2:

Input Error

输入3:

10000

输出3:

ans4 = 10000
ans6 = 10000

输入4:

34567

输出4:

ans4 = 37654
ans6 = 57643

输入5:

48935

输出5:

ans4 = 39854

ans6 = 59843

输入6:

98765

输出6:

ans4 = 56789

ans6 = 59876

输入7:

99999

输出7:

ans4 = 99999

ans6 = 99999

输入8:

40000

输出8:

ans4 = 40000

ans6 = 40000

输入9:

60000

输出9:

ans4 = 60000

ans6 = 60000

输入10:

50000

输出10:

```
ans4 = 50000
```

```
ans6 = 50000
```