23程设I-周7-课后1

实时评测 2023-10-26 20:00 1000 ms 32 MB 饶波

题目

请实现一个函数allOddBits。它会返回1，当且仅当输入的二进制数字所有奇数位被设为了1。位的编号是从0到31编号的，其中，0代表最低位（least significant）31代表最高位（most significant）例如：allOddBits(0xFFFFFFFD) = 0, allOddBits(0xAAAAAAAA) = 1。输入样例为0xAAAAAAAA

bonus:
这道题可以不用使用for循环和if判断只使用位运算和逻辑运算完成

问题输入

一个数a，范围为int

问题输出

0或1

输入样例

```
-1431655766
```

输出样例

```
1
```

main.c

```c
#include <stdio.h>

//problem from cmu csapp datalab
int allOddBits(int x) {
  int mask = 0xaa;
  mask = mask << 8 | mask;
  mask = mask << 8 | mask;
  mask = mask << 8 | mask;
  return !((x & mask) ^ mask);
}

int main() {
    int a;
    scanf("%d", &a);
    printf("%d", allOddBits(a));
}
```

23程设I-周9-课后3

实时评测 2023-11-02 20:00 1000 ms 32 MB 李磊鑫 (lilx58@mai2.sysu.edu.cn)

题目

给定一个时间x，以及一个非负整数表示的经过的分钟数m。你需要输出一个时间y，满足y过了m分钟变为x。

输入格式

输入两个数字x,m，x代表时间，由四个字符组成，如`2359`代表晚上23点59分，`0000`代表0点0分。m代表分钟数，是一个正整数。
1<=m<=1000
保证x是个有效时间

输出格式:

输出满足的时间y。

输入样例1:

```
1130 120
```

输出样例1:

```
0930
```

输入样例2:

```
0000 1
```

输出样例2:

```
2359
```

提示：可以使用char来读取前四个数字，并计算出时间。

1.c

```c
int main() {
    char h1,h2,m1,m2;
    int t;
    scanf("%c%c%c%c%d",&h1,&h2,&m1,&m2,&t);
    int hour = (h1-'0')*10+(h2-'0');
    int minute = (m1-'0')*10+(m2-'0');
    int total_minutes = hour * 60 + minute - t;
    if (total_minutes < 0) {
        total_minutes += 1440;  // 24 hours in minutes
    }
    printf("%02d%02d\n", total_minutes / 60, total_minutes % 60);
    return 0;
}
```

23程设I-周9-课后4

实时评测 2023-11-02 20:00 1000 ms 32 MB 李磊鑫 (lilx58@mai2.sysu.edu.cn)

题目

编写一个程序，接受用户输入的日期，包括年份、月份和日，然后判断输入的日期是否合法。日期应在1月1日和12月31日之间，考虑闰年。闰年通常规则是年份可被4整除，但不能被100整除，或者可以被400整除。

输入格式

输入应为三个整数，分别表示年份、月份和日期，用空格分隔。

输出格式

如果输入的日期合法，输出 "日期合法"；如果输入的日期不合法，输出 "日期不合法"。

输入样例1

```
2023 5 15
```

输出样例1

```
日期合法
```

输入样例2

```
2024 2 30
```

输出样例2

```
日期不合法
```

```c
#include <stdio.h>

int main() {
    int year, month, day;
    scanf("%d %d %d", &year, &month, &day);
    int isLeapYear = 0;  // 默认不是闰年
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        isLeapYear = 1;  // 如果年份为闰年，标记为闰年
    }
    int valid = 1;  // 默认日期合法
    switch (month) {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            if (day < 1 || day > 31) {
                valid = 0;
            }
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            if (day < 1 || day > 30) {
                valid = 0;
            }
            break;
        case 2:
            if (isLeapYear) {
                if (day < 1 || day > 29) {
                    valid = 0;
                }
            } else {
                if (day < 1 || day > 28) {
                    valid = 0;
                }
            }
            break;
        default:
            valid = 0;
            break;
    }
    if (valid) {
        printf("日期合法\n");
    } else {
        printf("日期不合法\n");
    }


    return 0;
}

```

23程设I-周10-课后2

实时评测 2023-11-10 20:00 1000 ms 32 MB 刘克钊 (liukzh9@mail2.sysu.edu.cn)

题目

可莉打算把星落湖里的鱼全部做成烤鱼！

星落湖初始有$N(0 < N \leq 100000)$条活鱼，可莉每次把湖里一半（向上取整）的活鱼做（炸）成烤鱼，直到湖里没有活鱼的时候才收手，请问可莉每次操作新获得的烤鱼数。

输入

鱼的初始数量N

输出

每次可以获得的烤鱼的数量，用空格隔开

样例1

```
1
```

样例1输出

```
1
```

样例2

```
10
```

样例2输出

```
5 3 1 1
```

先制成烤鱼5条，剩5条；再制成烤鱼3条，剩2条；再制成烤鱼1条，剩下1条；再制成烤鱼1条，湖中没有剩余的活鱼。

注意

可以使用math.h头文件中的floor(n)/ceil(n)向下/上取整，需要注意floor(n)/ceil(n)返回双精度浮点数

```
double pi = 3.14;
printf("pi向下取整为:%d, 向上取整为:%d", (int)floor(pi), (int)ceil(pi)); //输出3,4
```

本题行末是否有多余空格不影响结果判定

main.c

```c
#include<stdio.h>
#include<math.h>
int main(){
    int n;
    scanf("%d", &n);
    while(n>0){
        printf("%d ", (int)ceil(n / 2.0));
        n = floor(n / 2.0);
    }
    return 0;
}
```

☆ 23程设I-周12-课后1

实时评测 2023-11-23 20:00 5000 ms 1000 MB matrix (notify@matrix.moe)

Description

小猫咪玩游戏，现在有n张牌依次放在地上，牌上有数字，数字的范围是-20~20，正数表示可以奖励对应数目的猫粮，负数表示减去对应数量猫粮。猫咪可以从中连续选择k张牌， 0<=k<=n（k=0表示不选牌，即0包猫粮），即能获得数量为k张牌的数字之和的猫粮。
实现一个函数，求出猫咪最多能获取几包猫粮，1<=数组长度<=100。已提供main函数解决输入输出，只需要实现函数即可

Input

n+1行
第一行一个数字，数组长度n。
接下来的n行输入数组的数字

Output

输出猫咪所能获得的最多猫粮数量

Sample Input

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

Sample Output

```
15
```

header.c header.h main.c

```c
#include "header.h"
int maxFood(int a[], int len){
    int ans = 0;
    int temp = 0;
    int i;
    for(i=0; i<len; i++){
        temp = (temp+a[i])> 0?(temp+a[i]):0;
        ans = ans>temp?ans:temp;
    }
    return ans;
}
```

☆ 23程设I-周12-课后2

实时评测 2023-11-23 20:00 1000 ms 32 MB matrix (notify@matrix.moe)

题目

给定一个m x n大小的整数矩阵（m行，n列），按顺时针螺旋的顺序打印矩阵中的所有元素。

例如一个3 x 3 的 矩阵：

1 2 3

4 5 6

7 8 9

输出应为:

1 2 3 6 9 8 7 4 5

数据规模

`1 <= n,m <= 10`，矩阵中任意元素都满足 `|val|<=100`。

输入样例1

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

输出样例1

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

输入样例2

```
1 1
1
```

输出样例2

```
1
```

提示

只需完成函数 void spiralOrder(int **matrix,int m,int n)，该函数参数matrix是一个二维矩阵，用二维数组进行存储，可以使用matrix[0][0]在该函数中访问矩阵第一行第一列的数据，m表示矩阵的行数，n表示矩阵的列数。该函数需要按照题意顺时针螺旋打印矩阵内容。

solution.c main.c solution.h

```c
#include<stdio.h>
int min(int a, int b)
{
    if(a<b)
    return a;
    else
    return b;
}
void spiralOrder(int **matrix, int m, int n)
{
        int left=0,right=n,up=0,down=m;
        for(int qq=0;qq<(min(m,n)+1)/2;++qq){
            for(int i=left;i<right;++i){//左右
                printf("%d ", matrix[up][i]);
            }
            for(int j=up+1;j<down;++j){//上下
                printf("%d ", matrix[j][right-1]);
            }
            if(down-up!=1)for(int k=right-2;k>=left;--k){//右左
                printf("%d ", matrix[down-1][k]);
            }
            if(right-left!=1)for(int l=down-2;l>up;--l){//下上
                printf("%d ", matrix[l][left]);
            }
            left++;right--;up++;down--;
        }
        printf("\n");
        return;
}


```

23程设I-周12-课后3

实时评测 2023-11-23 20:00 1000 ms 32 MB matrix (notify@matrix.moe)

Description

小G在玩一个游戏，利用字符重复出现的次数，编写一种方法，实现基本的字符串压缩功能。比如，字符串aabccccccaaa会变为a2b1c5a3，假如压缩后的字符串的长度不小于原字符串的长度，则输出原来的字符串。
字符串中只包含小写英文字母（a至z），每个字母的重复次数为1到9次，字符总长度为1~100。

Hint

strlen()可以用来求一个字符串的长度，需要include头文件<string.h>

Input

一行，字符串

Output

输出压缩的字符串或者原字符串

Sample Input

```
aabccccccaaa
```

Sample Output

```
a2b1c5a3
```

Sample Input2

```
abbccd
```

Sample Output2

```
abbccd
```

(因为压缩后的字符串为a1b2c2d1，长度比原字符串abbccd大，所以输出原字符串)

main.c

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char s[105];
    scanf("%s", s);
    char ans[210];
    char ch = s[0];
    int temp = 1;
    int j = 0;
    for(int i=1; i<strlen(s); i++){
        if(ch == s[i]){
            temp++;
        }
        else{
            ans[j++] = ch;
            ans[j++] = (temp+'0');
            ch = s[i];
            temp = 1;
        }
    }
    ans[j++] = ch;
    ans[j++] = (temp+'0');
    ans[j] = '\0';
    if(j>=strlen(s))
    {
        printf("%s\n", s);
    }
    else {
        printf("%s\n", ans);
    }
    return 0;
}


```

23程设I-周12-课后4

实时评测 2023-11-23 20:00 1000 ms 32 MB matrix (notify@matrix.moe)

题目

输入两个非负整数a和b，输出两个非负整数的和(a+b)。

数据规模

0<=**a,b**<= $10^{99}-1$

main.c

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

void Inverse(char *s)
{
    int i;
    int length;
    char t;
    length=strlen(s);
    for(i=0;i<length-i-1;i++)
    {
        t=s[i];
        s[i]=s[length-i-1];
        s[length-i-1]=t;
    }
}

int Sum(char* a,char* b,char* sum)
{
    int m=strlen(a);
    int n=strlen(b);
    int acc = 0;
    int t,i;
    Inverse(a);
    Inverse(b);
    for(i = 0;i < m || i < n;i++)
    {
        if(i >= m)
            t = b[i] - '0' + acc;
        else if(i >= n)
            t = a[i] - '0'+ acc;
        else
            t = a[i] - '0'+ b[i] - '0' + acc;
        sum[i] = t % 10 + '0';
        if(t > 9)
            acc = 1;
        else
            acc = 0;
    }
    if(acc == 1)
        sum[i++] = '1';
    sum[i]='\0';
    Inverse(sum);
}
int main()
{
    char a[100],b[100],sum[250];
    scanf("%s%s",&a,&b);
    Sum(a,b,sum);
    printf("%s\n",sum);
```

```
51      return 0;
52  }
```

☆ 23程设I-周13-课后4

实时评测 2023-11-30 20:00 1000 ms 32 MB 崔亚群 (cuiyq6@mail2.sysu.edu.cn)

Description

现在给定两条字符串，如“abcdabc”和"abc"，要求字符串"abc"在"abcdabc"的最大字符串长度。最大字符串长度指的是"abc"在"abcdabc"连续出现的长度，当匹配的次数大于等于1的时候，不要求"abc"在后面也要保持完整。如"abcab"和“abc”的最大字符串长度为5。
字符串长度不大于10000。

Input

输入的第一个数字 N 代表测试样例数目，接下来有 $2 * N$ 行字符串。
每两行字符串的第一个字符串为待匹配的字符串，第二个字符串为要匹配的字符串。

Output

题意要求的最大字符串匹配长度。

Sample Input

```
2
abcdabc
abc
abcdabcd
abcd
```

Sample Output

```
3
8
```

main.c

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

char s[100000], m[100000];
int main(void){
    int t;
    scanf("%d", &t);
    while(t--){
        scanf("%s", s, 100000);
        scanf("%s", m, 100000);
        int s_len = strlen(s);
        int m_len = strlen(m);

        int i, j, max = 0;
        for(i = 0; i < s_len; i++){
            int not_match = 1;
            for(j = 0; j < s_len - i; j++){
                if(s[i + j] != m[j % m_len]){
                    not_match = 0;
                    break;
                }
            }
            if(j / m_len && max < j){
                max = j;
            }
        }

        printf("%d\n", max);
    }

    return 0;
}

```

23程设I-周15-课后1

实时评测 2023-12-14 20:00 1000 ms 32 MB 邱荣邦 (qiurb@mail2.sysu.edu.cn)

题目描述

根据主函数和头文件提示，编写子函数

```
void matrixInput(int (*mat)[COL]);
void matrixPrint(int *mat[ROW]);
void matrixAddT(int *mat);
```

分别实现矩阵的**输入**，**输出**，**与自身的转置相加**：

1. `void matrixInput(int (*mat)[COL]);`： 输入 `int` 类型的矩阵元素，用方阵 `mat` 存储；
2. `void matrixPrint(int *mat[ROW]);` ：输出方阵 `mat` 中的各个元素；
3. `void matrixAddT(int *mat);` ：计算方阵 `mat` 与自身转置的和，结果存储在 `mat` 中。

Hint：请注意三个子函数的传入参数类型**各不相同**。

输入格式

输入一个**三行三列**的方阵 A，每个元素均为 `int` 类型。

输出格式

输出一个**三行三列**的方阵 B，每个元素均为 `int` 类型，$B = A + A^T$。

matrixAddT.c main.c matrixAddT.h

```c
#include <stdio.h>
#include "matrixAddT.h"

void matrixInput(int (*mat)[COL])
{
    int row, col;
    for (row = 0; row < ROW; row++)
    {
        for (col = 0; col < COL; col++)
        {
            scanf("%d", *(mat + row) + col);
        }
    }
}


void matrixPrint(int *mat[ROW])
{
    int row, col;
    for (row = 0; row < ROW; row++)
    {
        for (col = 0; col < COL; col++)
        {
            printf("%d ", *(*(mat + row) + col));
        }
        printf("\n");
    }
}


void matrixAddT(int *mat)
{
    int row, col;
    for (row = 0; row < ROW; row++)
    {
        for (col = row; col < COL; col++)
        {
            int tmp = *(mat + row * ROW + col) + *(mat + col * COL + row);
            *(mat + row * ROW + col) = tmp;
            *(mat + col * COL + row) = tmp;
        }
    }
}
```

题目描述

请根据主函数和头文件提示，编写子函数

```
void merge(int *arr, int start, int mid, int end);
void mergeSort(int *arr, int start, int end);
```

以实现归并排序。

1. `void merge(int *arr, int start, int mid, int end);` ：将两个已排序的子数组进行合并，第一个子数组从下标 `start` 开始到下标 `mid` 结束，第二个子数组从下标 `mid+1` 开始到下标 `end` 结束。
2. `void mergeSort(int *arr, int start, int end);`： 输入数组 `arr` 、起始下标 `start` 和结束下标 `end` ，将 `arr` 中从下标 `start` 到下标 `end` 的子数组进行排序。

Hint：`void mergeSort(int *arr, int start, int end);` 是一个递归函数，当 `start<end` 时，会计算 `mid=(start+end)/2` ，然后分别调用 `mergeSort(arr, start, mid)` 和 `mergeSort(arr, mid+1, end)` 将数组分成两个子数组分别排序，之后再调用 `merge(arr, start, mid, end)` 将两个有序的子数组合并成一个有序的数组。

输入格式

第一行输入一个 `int` 类型变量 n，表示输入的数组长度，0<n<=10000;
第二行输入 n 个 `int` 类型的数组元素。

输出格式

按**从小到大**排好序的数组。

mergeSort.c main.c mergeSort.h

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

void merge(int *arr, int start, int mid, int end)
{
    int *result = (int*) malloc(sizeof(int) * (end - start + 1));
    int i = start, j = mid+1, k = 0;
    while(i <= mid && j <= end)
    {
        if(*(arr + i) > *(arr + j))
        {
            *(result + k) = *(arr + j);
            j++;
        }
        else
        {
            *(result + k) = *(arr + i);
            i++;
        }
        k++;
    }
    if(i > mid)
    {
        for(; j <= end; j++, k++)
        {
            *(result + k) = *(arr + j);
        }
    }
    if(j > end)
    {
        for(; i <= mid; i++, k++)
        {
            *(result + k) = *(arr + i);
        }
    }
    for(i = start; i <= end; i++)
    {
        *(arr + i) = *(result + i - start);
    }
    free(result);
}

void mergeSort(int *arr, int start, int end)
{
    if (start < end)
    {
        int mid = (start + end) / 2;
        mergeSort(arr, start, mid);
        mergeSort(arr, mid + 1, end);
```

```
51          merge(arr, start, mid, end);
52      }
53  }
```

☆ 23程设I-B-周17-课堂1

实时评测 2023-12-22 18:10 1000 ms 32 MB 邹广星 (wugx27@mail2.sysu.edu.cn)

题目描述

请你定义一个结构体表示课程信息，它包括课程名、上课时间和下课时间。现在你要编写一个程序，输入当天的所有课程，然后**按照上课时间从早到晚**的顺序输出当天课表。

输入输出描述

输入：第一行为当天课程总数 `n`，接下来 `n` 行每行有三个值 `class_name`、`start_time`、`end_time` 分别表示课程名、上课时间和下课时间。

输出：按照上课时间从早到晚的顺序输出当天课表，输出格式类似 `08:00 ~ 09:00 : C_Programing`，如当天没有课程则输出 `QiDong`

```
// printf的格式化字符串输出可参考:
printf("%02d:00 ~ %02d:00 : %s\n", start_time, end_time, class_name);
```

- `n` 范围为 `[0, 20]`
- `class_name` 是只包含大小写字母以及下划线的字符串，长度范围为 `[1, 15]`
- 上下课时间都是整数，且 $6 \leq$ `start_time` $<$ `end_time` ≤ 22
- 保证每门课的上课时间两两互不相同

样例输入 1

```
5
C_Programing 10 12
English 7 9
Physics 13 15
Calculus 18 19
PE 16 17
```

样例输出 1

```
07:00 ~ 09:00 : English
10:00 ~ 12:00 : C_Programing
13:00 ~ 15:00 : Physics
16:00 ~ 17:00 : PE
18:00 ~ 19:00 : Calculus
```

main.c

```c
#include <stdio.h>

struct node {
    char class_name[20];
    int start;
    int end;
};

void swap(struct node* a, struct node* b) {
    struct node tmp = *a;
    *a = *b;
    *b = tmp;
}

void sort(struct node classes[], int n) {
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            if (classes[i].start > classes[j].start)
                swap(&classes[i], &classes[j]);
        }
    }
}

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

    if (n == 0) {
        puts("QiDong");
        return 0;
    }

    struct node classes[50];
    for (int i = 0; i < n; ++i)
        scanf("%s%d%d", classes[i].class_name, &classes[i].start, &classes[i].end);

    sort(classes, n);

    for (int i = 0; i < n; ++i)
        printf("%02d:00 ~ %02d:00 : %s\n", classes[i].start, classes[i].end, classes[i].class_name);

    return 0;
}
```

☆ 23程设I-B-周17-课堂2

实时评测 2023-12-22 18:10 1000 ms 32 MB 邹广星 (wugx27@mail2.sysu.edu.cn)

题目描述

请你实现一个函数，它将删除一个链表中节点值为 `value` 的节点（*如果有的话*），并将该节点的前后节点（*如果有的话*）拼接起来以保持链表的连贯性，最后返回链表头，如删除后链表为空则返回 `NULL`。

`ListNode.h` 文件中已经给出链表节点 `ListNode` 以及函数 `deleteNodeOfList` 的声明，请在 `ListNode.c` 中实现该函数。

输入输出描述

（你无需处理输入和输出，只需要实现删除函数即可）

输入：第一行输入两个数 `n value` 分别代表链表的节点数和需要删除的值，第二行输入 `n` 个数表示这个链表的每个节点值

输出：删除后的链表的每个节点值，如果链表为空则输出 `NULL`

- 链表节点数 `n` 范围为 `[0, 50]`
- `value` 以及链表的每个节点的值 `val` 范围为 `[-100, 100]`
- 保证链表内的节点值**两两互不相同**

样例输入 1

```
4 2
1 4 2 3
```

样例输出 1

```
1 4 3
```

ListNode.c ListNode.h main.c

```c
#include "ListNode.h"

struct ListNode* deleteNodeOfList(struct ListNode* list, int value) {
    if (list == NULL)   return list;
    struct ListNode* head = list, *pre = list;
    if (head->val == value) {
        head = head->next;
        free(pre);
        return head;
    }

    struct ListNode* cur = pre->next;
    while (cur) {
        if (cur->val == value) {
            pre->next = cur->next;
            free(cur);
            break;
        }
        pre = cur;
        cur = cur->next;
    }
    return head;
}
```

☆ 23程设I-周17-课后1

实时评测 2023-12-28 20:00 1000 ms 32 MB 邹广星 (wugx27@mail2.sysu.edu.cn)

题目描述

请你实现一个函数，它将两个**升序**链表合并为一个新的**升序**链表**并返回链表头**（**升序**在这里指的是：前一个节点值 ≤ 后一个节点值）。新链表是通过有序拼接给定的两个链表的所有节点组成的。

`ListNode.h` 文件中已经给出链表节点 `ListNode` 以及合并函数 `mergeTwoLists` 的声明，请在 `ListNode.c` 中实现该函数。（*请尽可能不要创建新的节点*）

输入输出描述

（*你无需处理输入和输出，只需要实现合并函数即可*）

输入：第一行为两个链表的节点个数 `n m`，接下来两行分别输入 `n` 个数和 `m` 个数表示这两个链表的节点值

输出：合并后的链表节点值，如果链表为空则输出 `NULL`

- 两个链表节点数范围在 `[0, 50]`
- 链表的每个节点的值 `val` 范围为 `[-100, 100]`
- 链表内的节点值均按**升序**排列

样例输入 1

```
4 2
1 2 4 4
3 4
```

样例输出 1

```
1 2 3 4 4 4
```

ListNode.c ListNode.h main.c

```c
#include "ListNode.h"

// 递归版本，比较简洁
struct ListNode* mergeTwoLists(struct ListNode* list1, struct ListNode* list2) {
    if (list1 == NULL)   return list2;
    if (list2 == NULL)   return list1;

    if (list1->val < list2->val) {
        list1->next = mergeTwoLists(list1->next, list2);
        return list1;
    } else {
        list2->next = mergeTwoLists(list1, list2->next);
        return list2;
    }
}

/*
循环版本，且不开新结点
struct ListNode* mergeTwoLists(struct ListNode* list1, struct ListNode* list2) {
    if (list1 == NULL)   return list2;
    if (list2 == NULL)   return list1;

    struct ListNode *head1 = list1, *head2 = list2, *cur_head = NULL;
    if (head1->val < head2->val) {
        cur_head = head1;
        head1 = head1->next;
    } else {
        cur_head = head2;
        head2 = head2->next;
    }
    struct ListNode *final_head = cur_head;

    while (head1 && head2) {
        if (head1->val < head2->val) {
            cur_head->next = head1;
            head1 = head1->next;
        } else {
            cur_head->next = head2;
            head2 = head2->next;
        }
        cur_head = cur_head->next;
    }

    cur_head->next = head1 ? head1 : head2;
    return final_head;
}
*/
```

23程设I-周17-课后2

实时评测 2023-12-28 20:00 1000 ms 32 MB 张领统 (zhanglt25@mail2.sysu.edu.cn)

题目描述

实现一个链表及相关函数，链表节点的下标从 **0** 开始，需要实现的函数如下：

- `int getValue(struct ListNode* head, int index);` 获取链表下标为 `index` 的节点的值，若下标无效返回 `0`。
- `int addAtIndex(struct ListNode* head, int index, int val);` 将一个值为 `val` 的节点插入到下标为 `index` 的节点前。若下标等于链表长度，则将节点插入链表尾。如下标大于链表长度，则插入失败。插入成功返回 `1`，插入失败返回 `0`。
- `int deleteAtIndex(struct ListNode* head, int index);` 若存在下标为 `index` 的节点，删除该节点，否则删除失败。删除成功返回删除节点的值，删除失败返回 `0`。
- `void reverseList(struct ListNode* head);` 反转整个链表。
- `void reverseIndex(struct ListNode* head, int left, int right);` 反转从下标为 `left` 到下标为 `right` 的节点，包括这两个节点。若下标不合理，则不反转。保证 `left<=right`。
- `void printList(struct ListNode* head);` 打印链表，有三个节点的链表打印结果: `val1 -> val2 -> val3 -> null`，结尾有换行符。
- `void freeList(struct ListNode* head);` 释放链表内存。

链表反转部分尽量使用O(1)的辅助空间，不要借助数组实现。
保证链表中每个节点 `val` 大于 `0`。

输入描述

第一行输入正整数 `n`，代表程序需要执行的操作数。
接下来 `n` 行，每行包含字符串 `s` (操作种类)及操作需要的参数。

`1 <= n <= 1000`

样例输入

```
21
addAtIndex 0 1
addAtIndex 0 2
addAtIndex 3 2
getValue 2
getValue 1
deleteAtIndex 0
deleteAtIndex 1
addAtIndex 1 3
addAtIndex 2 4
addAtIndex 2 5
addAtIndex 1 8
reverseIndex 0 3
reverseIndex 1 3
reverseIndex 0 5
reverseList
getValue 2
getValue 5
deleteAtIndex 1
reverseIndex 2 3
reverseIndex 1 1
reverseList
```

样例输出

```
addAtIndex: add 1 at 0
addAtIndex: add 2 at 0
addAtIndex: fail
getValue: fail
getValue: get 1
deletaAtIndex: delete 2 at 0
deleteAtIndex: fail
addAtIndex: add 3 at 1
addAtIndex: add 4 at 2
addAtIndex: add 5 at 2
addAtIndex: add 8 at 1
5 -> 3 -> 8 -> 1 -> 4 -> null
5 -> 1 -> 8 -> 3 -> 4 -> null
5 -> 1 -> 8 -> 3 -> 4 -> null
4 -> 3 -> 8 -> 1 -> 5 -> null
getValue: get 8
getValue: fail
deletaAtIndex: delete 3 at 1
4 -> 8 -> 5 -> 1 -> null
4 -> 8 -> 5 -> 1 -> null
1 -> 5 -> 8 -> 4 -> null

1 -> 5 -> 8 -> 4 -> null
```

ListNode.c main.c ListNode.h

```c
#include <stdio.h>

#include "ListNode.h"

int getValue(struct ListNode* head, int index) {
    struct ListNode* curr = head->next;
    while (curr != NULL && index != 0) {
        curr = curr->next;
        index--;
    }

    return curr ? curr->val : 0;
}

int addAtIndex(struct ListNode* head, int index, int val) {
    struct ListNode* prev = head;
    struct ListNode* curr = head->next;

    while(curr != NULL && index != 0) {
        prev = curr;
        curr = curr -> next;
        index--;
    }

    if (curr == NULL && index != 0) {
        return 0;
    }

    struct ListNode* newNode = (struct ListNode*)malloc(sizeof(struct ListNode));
    newNode->val = val;
    prev->next = newNode;
    newNode->next = curr;
    return 1;
}

int deleteAtIndex(struct ListNode* head, int index) {
    struct ListNode* prev = head;
    struct ListNode* curr = head->next;

    while(curr != NULL && index != 0) {
        prev = curr;
        curr = curr -> next;
        index--;
    }

    if (curr == NULL) {
        return 0;
    }

    prev->next = curr->next;
```

```c
    int res = curr->val;
    free(curr);
    return res;
}

void reverseList(struct ListNode* head) {
    if (head->next == NULL) return;

    struct ListNode* prev = NULL;
    struct ListNode* curr = head->next;
    struct ListNode* last = curr->next;

    while(last != NULL) {
        curr->next = prev;
        prev = curr;
        curr = last;
        last = last->next;
    }
    curr->next = prev;
    head->next = curr;
}

void reverseIndex(struct ListNode* head, int left, int right) {
    struct ListNode* rightNode = head->next;
    while (rightNode != NULL && right != 0) {
        rightNode = rightNode->next;
        right--;
    }

    if (rightNode == NULL) return;

    struct ListNode* leftNode = head->next;
    struct ListNode* prev = head;
    while(leftNode != NULL && left != 0) {
        prev = leftNode;
        leftNode = leftNode->next;
        left--;
    }

    struct ListNode* last = rightNode->next;
    rightNode->next = NULL;

    reverseList(prev);
    leftNode->next = last;
}

void printList(struct ListNode* head) {
    while(head != NULL) {
        printf("%d -> ", head->val);
        head = head->next;
    }
    printf("null\n");
}

void freeList(struct ListNode* head) {
    while(head != NULL) {
        struct ListNode* curr = head;
        head = head->next;
        free(curr);
    }
}
```

23程设I-周17-课后3

实时评测 2023-12-28 20:00 1000 ms 32 MB 张领统 (zhanglt25@mail2.sysu.edu.cn)

题目描述

用链表实现一个栈及相关函数，需要实现的函数如下：

- `struct Stack* initStack();` 初始化一个栈，返回其指针。
- `int top(struct Stack* stack);` 获取栈顶元素，若栈为空，返回 `-1` 。
- `void push(struct Stack* stack, int val);` 插入一个元素。
- `int pop(struct Stack* stack);` 删除一个元素，返回元素值，若栈为空，返回 `-1` 。
- `int empty(struct Stack* stack);` 检查栈是否为空，若为空返回 `1` ，若不空返回 `0` 。
- `void freeStack(struct Stack* stack);` 释放栈的内存。

保证栈中每一个元素 `val` 都大于等于 `0` 。

输入描述

第一行输入正整数 `n` ，代表程序需要执行的操作数。
接下来 `n` 行，每行包含字符串 `s` (操作种类)及操作需要的参数。

`1 <= n <= 1000`

样例输入

```
15
top
empty
push 2
push 3
top
empty
pop
push 6
push 7
pop
pop
pop
pop
push 9
top
```

样例输出

```
top: fail
stack empty: true
push: 2
push: 3
top: 3
stack empty: false
pop: 3
push: 6
push: 7
pop: 7
pop: 6
pop: 2
pop: fail
push: 9
top: 9
```

提示

栈 -- 先进后出

stack.c main.c stack.h

```c
#include "stack.h"

struct Stack* initStack() {
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    struct ListNode* head = (struct ListNode*)malloc(sizeof(struct ListNode));
    head->next = NULL;
    stack->head = head;
    return stack;
}

int top(struct Stack* stack) {
    if (empty(stack)) return -1;
    struct ListNode* head = stack->head;
    return head->next->val;
}

void push(struct Stack* stack, int val) {
    struct ListNode* head = stack->head;
    struct ListNode* newNode = (struct ListNode*)malloc(sizeof(struct ListNode));
    newNode->val = val;
    newNode->next = head->next;
    head->next = newNode;
}

int pop(struct Stack* stack) {
    if (empty(stack)) return -1;
    struct ListNode* head = stack->head;
    struct ListNode* curr = head->next;
    int res = curr->val;
    head->next = curr->next;
    free(curr);
    return res;
}

int empty(struct Stack* stack) {
    struct ListNode* head = stack->head;
    return head->next == NULL;
}

void freeStack(struct Stack* stack) {
    struct ListNode* head = stack->head;
    while (head != NULL) {
        struct ListNode* curr = head;
        head = head->next;
        free(curr);
    }
    free(stack);
}
```

23程设I-周17-课后4

实时评测 2023-12-28 20:00 1000 ms 32 MB 张领统 (zhanglt25@mail2.sysu.edu.cn)

题目描述

请对链表排序问题使用归并排序吧！

从小到大排序，尽量不使用数组而在原链表上进行排序。

输入描述

第一行为正整数 `n` ，代表链表节点个数。 `0 <= n <= 200000`
第二行为 `n` 个数，代表链表节点的数值。

样例输入

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

样例输出

```
1 -> 2 -> 3 -> 4 -> 5 -> null
```

提示

归并排序的实现可能需要使用到以下知识：

1. 递归
2. 链表拆分 -- 快慢指针
3. 链表合并 -- 尽量使用迭代(递归可能会爆栈)

sort.c sort.h main.c

```c
#include "sort.h"

struct ListNode* mergeSort(struct ListNode* head) {
    if (head == NULL || head->next == NULL) return head;

    struct ListNode* prev = NULL;
    struct ListNode* slow = head;
    struct ListNode* fast = head;

    while (fast != NULL && fast->next != NULL) {
        prev = slow;
        slow = slow->next;
        fast = fast->next->next;
    }

    prev->next = NULL;

    struct ListNode* list1 = mergeSort(head);
    struct ListNode* list2 = mergeSort(slow);

    return merge(list1, list2);
}

struct ListNode* merge(struct ListNode* list1, struct ListNode* list2) {
    struct ListNode* head = (struct ListNode*)malloc(sizeof(struct ListNode));
    head->next = NULL;

    struct ListNode* curr = head;
    while (list1 != NULL || list2 != NULL) {
        if (list1 == NULL) {
            curr->next = list2;
            curr = curr->next;
            list2 = list2->next;
        } else if (list2 == NULL) {
            curr->next = list1;
            curr = curr->next;
            list1 = list1->next;
        } else {
            int val1 = list1->val;
            int val2 = list2->val;
            if (val1 < val2) {
                curr->next = list1;
                curr = curr->next;
                list1 = list1->next;
            } else {
                curr->next = list2;
                curr = curr->next;
                list2 = list2->next;
            }
        }
    }
    curr->next = NULL;

    struct ListNode* res = head->next;
    free(head);
    return res;
}
```

23程设I-B-周18-课堂1

实时评测 2023-12-29 18:25 1000 ms 32 MB 郑纪祥 (1782119136@qq.com)

Description

之前大家学习过结构体的简单应用了。现在要开始学习结构体的一些高级应用了——链表。

链表的定义依赖于下列结构体：

```
struct Node {
        struct Node* next;
        int value;
};
```

链表是依靠一个一个节点连接而成：

```
 _ _    _ _    _ _
|v|n|->|v|n|->|v|n|->
```

其中`v`代表`value`，`n`存储着下一个节点的地址。

给出一串数字，大家要有序地插入到链表中。举个例子：`5 1 3 4`在链表中的顺序为`1 3 4 5`。

函数的声明已经给出，请大家补全。

```
void insert(struct Node** head, int num);

void print_linklist(struct Node* head);

void delete_linklist(struct Node* head);
```

其中`insert`是用来插入元素到链表中，`print_linklist`是从链表头输出到链表尾，`delete_linklist`是删除链表。

输入的第一行是数字个数 N。第二行是该串数字。输出链表的元素。$1 <= N <= 1000$

Sample Input

```
4
5 1 3 4
```

Sample Output

```
1 3 4 5
```

fun.c fun.h main.c

```c
#include "fun.h"

void insert(struct Node** head, int num) {
    struct Node* pre = NULL, *cur = *head;
    struct Node* t = (struct Node*) malloc(sizeof(struct Node));
    t->next = NULL;
    t->value = num;
    while (cur != NULL && num >= cur->value) {
        pre = cur;
        cur = cur->next;
    }
    if (pre == NULL) {
        t->next = *head;
        *head = t;
    }
    else {
        pre->next = t;
        t->next = cur;
    }
    pre = NULL;
    cur = NULL;
}

void print_linklist(struct Node* head) {
    struct Node* p = head;
    printf("%d", p->value);
    p = head->next;
    while (p != NULL) {
        printf(" %d", p->value);
        p = p->next;
    }
    printf("\n");
}

void delete_linklist(struct Node* head) {
    struct Node* p = NULL;
    while (head != NULL) {
        p = head;
        head = head->next;
        free(p);
    }
    p = NULL;
}
```

23程设I-B-周18-课堂2

实时评测 2023-12-29 18:10 1000 ms 32 MB 郑纪祥 (1782119136@qq.com)

Descrption

1只公鸡值5文钱；1只母鸡值3文钱；3只小鸡值1文钱。请问用100文钱买100只鸡，公鸡、母鸡和小鸡各有几只?

实际题目中会按照M文钱买N只鸡的形式, 0 < M, N <= 250

按[公鸡、母鸡、小鸡]的顺序分别输出结果，一组解答占一行，解答按照公鸡数目从大到小排序（其次母鸡，再次小鸡）

无解时请输出 no answer

注意：钱要花完

Sample Input 1

```
100 100
```

Sample Ouput 1

```
12 4 84
8 11 81
4 18 78
0 25 75
```

Sample Input 2

```
1 4
```

Sample Output 2

```
no answer
```

main.c

```c
#include<stdio.h>
int main() {
    int a, b, c;
    int money, number, flag;
    scanf("%d %d", &money, &number);
    flag = 0;
    for (a = money/5; a >= 0; a--)
        for (b = money/3; b >= 0; b--)
            for (c = money; c >= 0; c--) {
                if (a + b + c*3 == number && a*5 + b*3 + c == money) {
                    flag++;
                    printf("%d %d %d\n", a, b, c*3);
                }
            }
    if ( flag == 0 )
        printf("no answer\n");
    return 0;
}
```

main.c

```c
#include<stdio.h>

int main() {
    int m, n, cnt = 0;
    scanf("%d %d", &m, &n);
    for (int i = m / 5; i >= 0; i--) {
        for (int j = (m - i * 5) / 3; j >= 0; j--) {
            int k = n - i - j;
            if (k >= 0 && (m - i * 5 - j * 3 - k * (1.0 / 3)) == 0) {
                printf("%d %d %d\n", i, j, k);
                cnt ++;
            }
        }
    }
    if (cnt == 0)
        printf("no answer\n");
    return 0;
}
```