Problem A. 佛像

难度	考点
1	转义字符的输出

题目分析

手动或者自动在`\`、`'`、`"`前添加`\`，在`%`前添加`%%`即可。

示例代码

```
#include <stdio.h>
int main()
{
    printf("                       _ooOoo_\n"
        "                      o8888888o\n"
        "                      88\" . \"88\n"
        "                      (| -_- |)\n"
        "                       O\\ = /O\n"
        "                   %%____/\'---\'\\____%%\n"
        "                     .\' \\\\| |// \'.\n"
        "                   / \\\\||| : |||// \\\n"
        "                 / _||||| -:- |||||- \\\n"
        "                   | | \\\\\\ - /// | |\n"
        "                 | \\_| \'\'\\---/\'\' |_/\n"
        "                 \\ .-\\__ `-` ___/-. /\n"
        "               ___`. .\' /--.--\\ `. . __\n"
        "            .\"\" \'< `.___\\_<|>_/___.\' >\'\"\".\n"
        "          | | : `- \\\`.;`\\ _ /`;.`/ - ` : | |\n"
        "            \\ \\ `-. \\_ __\\ /__ _/ .-` / /\n"
        "    %%%%=====`-.____`-.___\\_____/___.-`____.-\'=====%%%%\n"
        "                          `=---=\'");
    return 0;
}
```

自动AC机

手动添加或许有些麻烦，写一个自动添加`\`和`%`的代码或许效率更高。

```c
#include<stdio.h>
int main(){
    freopen("in.txt","r",stdin);
    freopen("ans.out","w",stdout);
    char c;
    while(~scanf("%c",&c)){
        if(c=='\\' || c=='\'' || c=='\"')
            putchar('\\');
        else if(c=='%')
            putchar(c);
        putchar(c);
    }
}
```

Problem B. 温度转换

难度	考点
1	循环、浮点数

题目分析

按题目所给公式进行温度转化即可，需要注意的只有出现过很多次的整数与浮点数之间的转换问题（即使用`9.0`、`5.0`参与计算）。

示例代码

```c
#include <stdio.h>
int main()
{
    int L, R;
    int i, op;
    scanf("%d%d%d", &op, &L, &R);
    for (i = L; i <= R; i++)
    {
        printf("%d %.2f\n", i, op == 0 ? 9.0 * i / 5 + 32 : 5.0 * (i - 32) / 9);
    }
}
```

Problem C. 基础物理实验

难度	考点
1	循环

题目分析

注意积分最多被扣到 0 分，因此迟到前扣分时需要判断一下当前分数是否小于 2 。

示例代码

```
#include<stdio.h>
int main(){
    int n,sum=0;
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        int id,op;
        scanf("%d%d",&id,&op);
        switch(op){
            case 0:sum+=id%10;break ;
            case 2:sum=sum<2?0:sum-2;
        }
    }
    if(sum>=37)
        printf("%d\n",sum);
    else puts("See you next year !");
}
```

Problem D. 子串逆置_PRO

难度	考点
3	字符串、指针

题目分析

S 中与 T 匹配的字串互不相交，且不用考虑逆置后会构成新的子串的情况。
注意可能会有回文串逆置后不变的情况，这并不算新的子串，因此**在成功匹配到符合条件的子串后，应该将指针后移继续匹配。**
`ababab`和`ab`匹配，第一次匹配的位置是第一和第二个字母，接下来应该从第三个字母开始继续往后匹配QAQ

示例代码

```
#include <stdio.h>
#include <string.h>
void rev(char *first, char *last)
{
    int tmp;
    while (first < last)
    {
        tmp = *last;
        *last = *first;
        *first = tmp;
        first++, last--;
    }
}
int main()
```

```
{
    char str[205], substr[250], *p, *now;
    scanf("%s%s", str, substr);
    p = str;
    while ((p = strstr(p, substr)) != NULL)
    {
        rev(p, p + strlen(substr) - 1);
        p += strlen(substr);
    }
    puts(str);
    return 0;
}
```

Problem E. Gcd-Expression

难度	考点
3	递归、指针、数论

题目分析

翻译一下伪代码即可，在调用函数前，定义变量 x、y，将它们的地址进行传参，递归的途中修改参数指针指向的值。

示例代码

```
#include<stdio.h>
int Exgcd(int a,int b,long long *x,long long *y){
    if(b==0){
        *x=1,*y=0;
        return a;
    }
    else{
        long long xx,yy;
        int d=Exgcd(b,a%b,&xx,&yy);
        *x=yy,*y=xx-(a/b)*yy;
        return d;
    }
}
int main(){
    int a,b;long long x,y;
    scanf("%d%d",&a,&b);
    int d=Exgcd(a,b,&x,&y);
    printf("%d = %d*(%lld) + %d*(%lld)\n",d,a,x,b,y);
}
```

Problem F. 成绩处理

难度	考点
5	多关键字排序、字符二维数组

题目分析

用全局变量`Score`数组和`Name`数组按输入顺序存储成绩和姓名，用数组`index[i]`表示排序后排名第i名的学生的数据输入的次序（也是`Score`数组和`Name`数组中的对应下标）。之后定义`qsort`函数中的`Cmp`函数，根据`Score`和`Name`数组的值，对`index`数组进行排序。排序过后，排名为1的同学的姓名为`Name[index[1]]`，其成绩为`Score[index[1]]`，依次输出即可。

对于`qsort`函数中需要自定义的`Cmp`函数，其传入参数为指向待排序数组中任意两个元素的指针。该函数根据两个传入的元素，返回`1`，`-1`，或`0`，表示传入元素的大小关系，决定了排序的方式。对于双关键字排序，可以使用示例程序中的方法编写`Cmp`函数。

需要注意的是，`Cmp`函数不应写成以下形式：

```
int Cmp(const void *pa, const void *pb)
{
    int a = *((int *)pa);
    int b = *((int *)pb);

    if (Score[a] != Score[b])
    {
        return Score[b] - Score[a];     // 小心减法溢出！
    }
    else
    {
        return strcmp(Name[a], Name[b]);
    }
}
```

虽然看起来类似，但变量减法操作容易导致溢出，使得一个负数减去一个正数得到一个正值，这会导致排序错误。本题中，成绩限制在$[0, 100]$内，故无需考虑此问题。

示例代码

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


int Score[1000 + 7];
char Name[1000 + 7][10 + 7];

int Cmp(const void *pa, const void *pb);

int main()
{
    int index[1000 + 7];
    int n;
    int i;
```

```
    scanf("%d", &n);
    for (i = 0; i < n; ++i)
    {
        scanf("\n%s%d", Name[i], &Score[i]);
        index[i] = i;
    }

    qsort(index, n, sizeof(int), Cmp);

    for (i = 0; i < n; ++i)
    {
        printf("%s %d\n", Name[index[i]], Score[index[i]]);
    }

    return 0;
}



int Cmp(const void *pa, const void *pb)
{
    int a = *((int *)pa);
    int b = *((int *)pb);

    if (Score[a] > Score[b])
    {
        return -1;
    }
    else if (Score[a] < Score[b])
    {
        return 1;
    }
    else
    {
        return strcmp(Name[a], Name[b]);
    }
}
```

Problem G. 划分测试集

难度	考点
3	哈希表

题目分析

注意到输入的数字范围介于$a_i \in [-100000, 100000]$之间，这提醒我们可以开一个大小超过200001的hash数组解决。令`ret`为出现的图片种类数。

在读入k个数字时，设当前读入的数字为a_i,若$hash[a_i]$为0，则代表之前并没有出现过该种类的图片，令`ret+1`,并使$hash[a_i]$为1，代表该数字出现过了。若$hash[a_i]$为1，则代表之前已经出现过该种类图片，因此不再令`ret+1`。

最后，若`ret>k/n`,则代表图片种类超过了测试集中图片个数，因此可以使测试集中图片种类数即为测试集中图片个数。若`ret<=k/n`，则表明测试集中最多可以有`ret`类图片。时间复杂度`O(n)`.

示例程序

```
#include<stdio.h>
long long n,k;
char hash[200005];
int main(void)
{
    int i,data,ret=0;
    scanf("%d %d",&n,&k);
    for(i=0;i<k;i++){
        scanf("%d",&data);
        data+=100000;
        if(!hash[data]){
            hash[data]=1;
            ret++;
        }
    }
    if(ret>k/n){
        ret=k/n;
    }
    printf("%d",ret);
    return 0;
}
```

Problem H. 错误的幻方

难度	考点
5	枚举、二维数组

题目分析

分为两种情况：

- 交换的两个数字在不同行不同列：一定会导致两行和两列的和产生错误。找到这两行两列，枚举其四个交点的交换方案即可。
- 交换的两个数字在同一行：会导致两列的和产生错误，找到这两列，枚举这两个数可能在的行，尝试交换并验证即可。
- 交换的两个数字在同一列：与情况2类似。

示例代码

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#define N 1005
```

```
int a[N][N],sumr[N],sumc[N],n,sum;
bool check(){
    for(int i=1;i<=n;i++){
        int sumr=0,sumc=0;
        for(int j=1;j<=n;j++)
            sumr+=a[i][j],sumc+=a[j][i];
        if(sumr!=sum) return 0;
        if(sumc!=sum) return 0;
    }
    return 1;
}
int main(){
    scanf("%d",&n),sum=(1+n*n)*n/2;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            scanf("%d",&a[i][j]);

    int row[5],r=0;
    int column[5],c=0;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            sumr[i]+=a[i][j];
            sumc[i]+=a[j][i];
        }
        if(sumr[i]!=sum) row[r++]=i;
        if(sumc[i]!=sum) column[c++]=i;
    }

    if(r==0){
        int y1=column[0],y2=column[1];
        for(int i=1;i<=n;i++){
            int s1=sumc[y1]-a[i][y1]+a[i][y2];
            int s2=sumc[y2]-a[i][y2]+a[i][y1];
            if(s1==sum && s2==sum){
                printf("%d %d\n%d %d\n",i,y1,i,y2);
                return 0;
            }
        }
    }

    else if(c==0){
        int x1=row[0],x2=row[1];
        for(int i=1;i<=n;i++){
            int s1=sumr[x1]-a[x1][i]+a[x2][i];
            int s2=sumr[x2]-a[x2][i]+a[x1][i];
            if(s1==sum && s2==sum){
                printf("%d %d\n%d %d\n",x1,i,x2,i);
                return 0;
            }
        }
    }

    else{
        int t=a[row[0]][column[0]];
        a[row[0]][column[0]]=a[row[1]][column[1]];
        a[row[1]][column[1]]=t;
```

```
        if(check())
            printf("%d %d\n%d %d\n",row[0],column[0],row[1],column[1]);
        else printf("%d %d\n%d %d\n",row[0],column[1],row[1],column[0]);
    }
}
```

Problem I. 很解压的字符串解压

难度	考点
5	指针、字符串处理、递归、栈

题目分析

通过递归调用模拟压栈和出栈

遍历字符串，这里可以利用字符指针处理，建议理解，争取掌握。

讨论当前字符可能的几种情况，并针对这些情况采取不同的操作：

- 字母：保存至结果数组中；
- 数字：更新计数值`num`；
- `(`：调用函数递归，将递归调用返回的结果复制`num`次到结果中；
- `)`：保存当前指针所在的位置，并返回结果。

  tips：这里用到了函数参数的**地址传递**，详细说明见如下链接：

  [值传递和地址传递，C语言函数传参方式详解 (biancheng.net)]

示例程序

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#define N 10010
char s[N];
char *read(char *p, char **e){                          //地址传递，参数e是指针的指针
    int cnt = 0, num = 0;
    char *ans = (char *)malloc(sizeof(char) * N);    //动态分配内存，否则REG
    char *buf, *end = NULL;
    while(*p){
        if(isalpha(*p))
            ans[cnt++] = *p;
        else if(isdigit(*p))
            num = 10 * num + *p - '0';
        else if(*p == '('){
            buf = read(p + 1, &end);
            while(num--){
                strcpy(ans + cnt, buf);
                cnt += strlen(buf);
```

```
            }
            num = 0;
            p = end;
        }
        else if(*p == ')'){
            *e = p;
            ans[cnt] = '\0';
            return ans;
        }
        p++;
    }
    ans[cnt] = '\0';
    return ans;
}
int main(){
    scanf("%s", s);
    char *p = s, *end = NULL;
    printf("%s", read(p, &end));
    return 0;
}
```

Problem J. MIPS

难度	考点
7	指针、模拟、字符串

题目分析

题目思路没什么好分析的，按要求模拟各个指令即可，需要注意的点有：

- 寄存器指向的位置可能储存了一个`int`，也可能储存了一个`int*`，因此需要`int *s[10]`和`int **s[10]`来模拟寄存器（以及仔细理解一下`lw`和`sw`指令，这应该是这道题的实现难点），以及不要用强制转换偷懒这个Hint里也说了QAQ。

  也注意处理`lw`和`sw`指令时，地址的偏转量需要除 4 。

  出题人的`lw`语句实现方法：

  ```
  int *t0=parameter(opt[i][0]);
  int **t1=_parameter(opt[i][1]);
  //parameter和_parameter函数是在根据输入字符串判断lw的参数是哪一个寄存器（比如是$s1还是$s2）
  *t0=*((*t1)+opt_num[i]/4);
  ```

- 输入当然可以用`gets`但由于有空格的分隔，个人觉得用`scanf("%s",)`会更方便。
- 关于标签跳转：使用`hash`记录每个标签位于多少行，遇见标签是强制跳转即可。
- 关于`.data`部分定义的数组怎么处理：或许可以动态定义，出题人的做法是静态定义了`Array_space[N+5][N+5]`，然后同样使用`hash`的方式，使每个数组名对应到某个`Array_space`。（静态的内存也会比较方便调试~）

- 看起来代码量比较大，但大多数的代码都是判断不同操作的`if`，想清楚了每个操作的指针模拟方法，这个题并不太困难~
- 以及所有数据来自真实的`mips`代码的运行结果（所以是不会有问题的.jpg）。

示例代码

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
#define N 105
#define M 1000007
#define key 97

int Array_id[M+5],Array_space[N+5][N+5];

int *s[10],*zero,*v0,*a0,Type[15];
int **_s[10],**_v0,**_a0;

int Line[M];

char name[N],type[N],str[N];
char opt[N][5][20],opt_type[N][10];
int opt_num[N];

int getint(){
    char c;int flag=0,num=0;
    while((c=getchar())<'0'||c>'9')if(c=='-')flag=1;
    while(c>='0'&&c<='9'){num=num*10+c-48;c=getchar();}
    return flag?-num:num;
}
int getid(char c){
    if(c>='A' && c<='Z')
        return c-'A'+1;
    else if(c>='a' && c<='z')
        return 26+c-'a'+1;
    else if(c>='0' && c<='9')
        return 52+c-'0'+1;
    else return 63;
}
int Hash(char *str){
    int ret=0;
    for(;*str!='\0';str++)
        ret=(ret*key+getid(*str))%M;
    return ret;
}
int* parameter(char *rig){
    if(rig[1]=='v')
        return v0;
    else if(rig[1]=='a')
        return a0;
    else if(rig[1]=='z')
        return zero;
    return s[rig[2]-48];
```

```
}
int** _parameter(char *rig){
    if(rig[1]=='v')
        return _v0;
    else if(rig[1]=='a')
        return _a0;
    return _s[rig[2]-48];
}
int ID(char *rig){
    if(rig[1]=='v')
        return 8;
    else if(rig[1]=='a')
        return 9;
    return rig[2]-48;
}
int main(){
    for(int i=0;i<8;i++){
        s[i]=(int*)malloc(4);
        _s[i]=(int**)malloc(sizeof(int*));
    }
    zero=(int*)malloc(4);
    v0=(int*)malloc(4);
    _v0=(int**)malloc(sizeof(int*));
    a0=(int*)malloc(4);
    _a0=(int**)malloc(sizeof(int*));

    *zero=0;
    gets(str);
    int num_array=0;
    while(~scanf("%s",str)){
        if(!strcmp(str,".text"))
            break ;

        int len=strlen(str);
        str[len-1]='\0';
        strcpy(name,str);
        scanf("%s",type);
        int x=getint(),h=Hash(name);
        Array_id[h]=++num_array;
        //printf("%s %s %d\n",name,type,SIZE);
        if(!strcmp(type,".word"))
            Array_space[num_array][0]=x;
    }

    int cnt=0;
    while(~scanf("%s",str)){
        int len=strlen(str);
        if(!len) continue ;

        cnt++;
        //printf("%d\n",cnt);
        if(str[len-1]==':'){
            strcpy(opt_type[cnt],"label");
            str[len-1]='\0';
            strcpy(opt[cnt][0],str);
            Line[Hash(str)]=cnt;
```

```
            continue ;
        }
        strcpy(opt_type[cnt],str);

        if(!strcmp(str,"li")){
            scanf("%s",opt[cnt][0]);
            opt_num[cnt]=getint();
        }
        else if((!strcmp(str,"move")) || (!strcmp(str,"la")))
            scanf("%s%s",opt[cnt][0],opt[cnt][1]);

        else if((!strcmp(str,"lw")) || (!strcmp(str,"sw"))){
            scanf("%s",opt[cnt][0]);
            opt_num[cnt]=getint();
            scanf("%s",opt[cnt][1]);
        }

        else if((!strcmp(str,"add")) || (!strcmp(str,"mul"))
                || (!strcmp(str,"beq")) || (!strcmp(str,"bne")))
            scanf("%s%s%s",opt[cnt][0],opt[cnt][1],opt[cnt][2]);

        else if(!strcmp(str,"addi")){
            scanf("%s%s",opt[cnt][0],opt[cnt][1]);
            opt_num[cnt]=getint();
        }

        else if(!strcmp(str,"j"))
            scanf("%s",opt[cnt][0]);
    }

    for(int i=1;i<=cnt;i++){
        if(!strcmp(opt_type[i],"label"))
            continue ;
        else if(!strcmp(opt_type[i],"j")){
            i=Line[Hash(opt[i][0])];continue ;
        }

        else if(!strcmp(opt_type[i],"li")){
            *(parameter(opt[i][0]))=opt_num[i];
            Type[ID(opt[i][0])]=0;
        }
        else if(!strcmp(opt_type[i],"move")){
            if(Type[ID(opt[i][1])]==0){
                int *t1=parameter(opt[i][0]);
                int *t2=parameter(opt[i][1]);
                *t1=*t2,Type[ID(opt[i][0])]=0;
            }
            else{
                int **t1=_parameter(opt[i][0]);
                int **t2=_parameter(opt[i][1]);
                *t1=*t2,Type[ID(opt[i][0])]=1;
            }
        }

        else if(!strcmp(opt_type[i],"la")){
            int id=Array_id[Hash(opt[i][1])];
```

```
            if(opt[i][0][1]=='v')
                (*_v0)=Array_space[id],Type[8]=1;
            else if(opt[i][0][1]=='a')
                (*_a0)=Array_space[id],Type[9]=1;
            else{
                (*_s[opt[i][0][2]-48])=Array_space[id];
                Type[opt[i][0][2]-48]=1;
            }
        }
        else if(!strcmp(opt_type[i],"lw")){
            int *t0=parameter(opt[i][0]);
            int **t1=_parameter(opt[i][1]);
            *t0=*((*t1)+opt_num[i]/4);
            Type[ID(opt[i][0])]=0;
        }
        else if(!strcmp(opt_type[i],"sw")){
            int *t0=parameter(opt[i][0]);
            int **t1=_parameter(opt[i][1]);
            *((*t1)+opt_num[i]/4)=*t0;
        }

        else if(!strcmp(opt_type[i],"add")){
            if(Type[ID(opt[i][1])]==0){
                int *t0=parameter(opt[i][0]);
                int *t1=parameter(opt[i][1]);
                int *t2=parameter(opt[i][2]);
                *t0=*t1+*t2;
                Type[ID(opt[i][0])]=0;
            }
            else{
                int **t0=_parameter(opt[i][0]);
                int **t1=_parameter(opt[i][1]);
                int *t2=parameter(opt[i][2]);
                *t0=*t1+(*t2)/4;
                Type[ID(opt[i][0])]=1;
            }
        }
        else if(!strcmp(opt_type[i],"addi")){
            if(Type[ID(opt[i][1])]==0){
                int *t0=parameter(opt[i][0]);
                int *t1=parameter(opt[i][1]);
                *t0=*t1+opt_num[i];
                Type[ID(opt[i][0])]=0;
            }
            else{
                int **t0=_parameter(opt[i][0]);
                int **t1=_parameter(opt[i][1]);
                *t0=*t1+opt_num[i]/4;
                Type[ID(opt[i][0])]=1;
            }
        }
        else if(!strcmp(opt_type[i],"mul")){
            int *t0=parameter(opt[i][0]);
            int *t1=parameter(opt[i][1]);
            int *t2=parameter(opt[i][2]);
            *t0=(*t1)*(*t2);
```

```
            Type[ID(opt[i][0])]=0;
        }

        else if(!strcmp(opt_type[i],"beq")){
            int Ty0=Type[ID(opt[i][0])];
            int Ty1=Type[ID(opt[i][1])];

            if(Ty0!=Ty1) continue ;
            else if(Ty0==0){
                int *t0=parameter(opt[i][0]);
                int *t1=parameter(opt[i][1]);
                if((*t0)==(*t1)){
                    i=Line[Hash(opt[i][2])];continue ;
                }
            }
            else{
                int **t0=_parameter(opt[i][0]);
                int **t1=_parameter(opt[i][1]);
                if((*t0)==(*t1)){
                    i=Line[Hash(opt[i][2])];continue ;
                }
            }
        }
        else if(!strcmp(opt_type[i],"bne")){
            int Ty0=Type[ID(opt[i][0])];
            int Ty1=Type[ID(opt[i][1])];

            if(Ty0!=Ty1){
                i=Line[Hash(opt[i][2])];continue ;
            }

            else if(Ty0==0){
                int *t0=parameter(opt[i][0]);
                int *t1=parameter(opt[i][1]);
                if((*t0)!=(*t1)){
                    i=Line[Hash(opt[i][2])];continue ;
                }
            }
            else if(Ty0==1){
                int **t0=_parameter(opt[i][0]);
                int **t1=_parameter(opt[i][1]);
                if((*t0)!=(*t1)){
                    i=Line[Hash(opt[i][2])];continue ;
                }
            }
        }

        else if(!strcmp(opt_type[i],"syscall")){
            if((*v0)==1)
                printf("%d",*a0);
            else if((*v0)==11)
                printf("%c",(*a0)&(0xff));
            else if((*v0)==10)
                return 0;
        }
    }
```

```
    return 0;
}
```