calc.c

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

void swap(char a[]){
	int L = strlen(a);
	int i;
	for (i = 0;i<L/2;i++){
		int t = a[i];
		a[i] = a[L-i-1];
		a[L-i-1] = t;
	}
}

void add(const char a[], const char b[], char res[]){
	int l1 = strlen(a);
	int l2 = strlen(b);
	int m;
	int n;
	int i;
	int num;

	if (l1==0 && l2==0){
		return;
	}

	if (l1>l2){
		m = l1;
		n = l2;
	}
	if (l1<=l2){
		m = l2;
		n = l1;
	}
	res[0] = '0';
	for (i=0;i<n;i++){
		num = res[i] - '0' + a[l1-1-i] - '0' + b[l2-1-i] - '0';
		res[i] = num%10 + '0';
		if (num>=10){
			res[i+1] = '1';
		}
		else res[i+1] = '0';
	}
```

```
        for (i=0;i<(m-n);i++){
            if (l1>l2){
                num = res[n+i] - '0' + a[m-n-1-i] - '0';
                res[n+i] = num%10 + '0';
                if (num>=10){
                    res[n+i+1] = '1';
                }
                else res[n+i+1] = '0';
            }

            if (l1<l2){
                num = res[n+i] - '0' + b[m-n-1-i] - '0';
                res[n+i] = num%10 + '0';
                if (num>=10){
                    res[n+i+1] = '1';
                }
                else res[n+i+1] = '0';
            }
        }

        swap(res);

        if (res[0] == '0'){
            int len = strlen(res);
            for (i=0;i<len-1;i++){
                res[i] = res[i+1];
            }
            res[len-1] = 0;
        }

}

void sub( const char a[], const char b[], char res []){
        int l1 = strlen(a);
        int l2 = strlen(b);
        int i = 0, j = 0;
        int m,n;
        int num;
        int num_a = 0, num_b = 0;
        int mark;
        int borrow;

        if (l1==0 && l2==0){
            return;
```

```
}

if (l1>l2){
	m = l1;
	n = l2;
}
if (l1<l2){
	m = l2;
	n = l1;
	mark = 1;
}
if (l1 == l2){
	m = n = l1;
	for (i=0;i<l1;i++){
		if (a[i]>b[i]){
			goto next;
		}
		if (a[i]<b[i]){
			mark = 1;
			goto next;
		}
		if (a[i] == b[i]){
			j++;
		}
	}
	if (j == l1){
		res[0] = '0';
		return;
	}
}

next:
if (mark != 1){
	for (i=0;i<n;i++){
		if (borrow){
			num = (a[l1-1-i] - '0') - (b[l2-1-i] - '0') - 1;
		}
		else{
			num = (a[l1-1-i] - '0') - (b[l2-1-i] - '0');
		}
		if (num<0){
			num += 10;
			borrow = 1;
		}
```

```
            else borrow = 0;

            res[i] = num + '0';
        }

        for (i=0;i<m-n;i++){
            if (borrow){
                num = a[m-n-1-i] - '0' - 1;
            }
            else{
                num = a[m-n-1-i] - '0';
            }
            if (num<0){
                num += 10;
                borrow = 1;
            }
            else{
                borrow = 0;
            }

            res[n+i] = num + '0';
        }

        int len = strlen(res);
        i=0;
        while (res[len-1-i] == '0'){
            res[len-1-i] = 0;
            i++;
        }
        swap(res);
    }

    if (mark == 1){
        for (i=0;i<n;i++){
            if (borrow){
                num = b[l2-1-i] - '0' - (a[l1-1-i] - '0') - 1;
            }
            else{
                num = b[l2-1-i] - '0' - (a[l1-1-i] - '0');
            }
            if (num<0){
                num += 10;
                borrow = 1;
            }
```

```
            else borrow = 0;

            res[i] = num + '0';
        }

        for (i=0;i<m-n;i++){
            if (borrow){
                num = b[m-n-1-i] - '0' - 1;
            }
            else{
                num = b[m-n-1-i] - '0';
            }
            if (num<0){
                num += 10;
                borrow = 1;
            }
            else borrow = 0;

            res[n+i] = num + '0';
        }

        int len = strlen(res);
        i=0;
        while (res[len-1-i] == '0'){
            res[len-1-i] = 0;
            i++;
        }
        len = strlen(res);
        res[len] = '-';
        swap(res);
    }

}
```

matrix.c

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

int is_same_diagonals(int n, int a[]){
    int i,j,k;
    int diagonals = 2*n - 3;
    int count = 0;
    int mark = 0;
```

```
//main diagonal
for (i=0;i<n-1;i++){
	if (a[i+i*n]==a[(i+1)+(i+1)*n]){
		count++;
	}
}
if (count==n-1){
	mark+=1;
}
else return 0;

//lower diagonals
for (k=2;k<n;k++){
	i=(k-1);
	j=(n-1);
	count = 0;
	while ((i>0)&&(j>((n-1)-k+1))){
		if (a[i+j*n]==a[(i-1)+(j-1)*n]){
			count++;
		}
		i--;
		j--;
	}
	if (count == k-1){
		mark+=1;
	}
	else return 0;
}

//upper diagonals
for (k=n-1;k>=2;k--){
	i=(n-1);
	j=(k-1);
	count = 0;

	while ((i>(n-1-k+1))&&(j>0)){
		if (a[i+j*n]==a[(i-1)+(j-1)*n]){
			count++;
		}
		i--;
		j--;
	}
	if (count == k-1){
		mark+=1;
```

```
            }
            else return 0;
        }

        if (mark==diagonals){
            return 1;
        }
        else return 0;
}
```

path.c

```
int recursion(int n, int m, int A[], int i, int j, int k, int l, int path[][2]){
        if ((A[i*m+j]==1) || (A[k*m+l==1])){
            return 0;
        }
        if ((i<0) || (i>=n)){
            return 0;
        }
        if ((j<0) || (j>=m)){
            return 0;
        }
        if (A[i*m+j]==9){
            return 0;
        }
        if ((i==k) && (j==l)){
            if ((A[i*m+j]==0) && (A[k*m+l]==0)){
                return 1;
            }
        }

        A[i*m+j]=9;

        if (recursion(n,m,A,i-1,j,k,l,path)){
            return 2;
        }
        else if (recursion(n,m,A,i+1,j,k,l,path)){
            return 2;
        }
        else if (recursion(n,m,A,i,j-1,k,l,path)){
            return 2;
        }
        else if (recursion(n,m,A,i,j+1,k,l,path)){
            return 2;
```

```
        }
        else{
                return 0;
        }

}

int find_path(int n, int m, int A[], int i, int j, int k, int l, int path[][2]){
        if ((A[i*m+j]==1) || (A[k*m+l==1])){
                return -1;
        }
        if ((i<0) || (i>=n)){
                return -2;
        }
        if ((j<0) || (j>=m)){
                return -3;
        }
        if ((i==k) && (j==l)){
                if ((A[i*m+j]==0) && (A[k*m+l]==0)){
                        return 1;
                }
        }

        int result = recursion(n,m,A,i,j,k,l,path);

        int x,y;
        for (x=0;x<n;x++){
                for (y=0;y<m;y++){
                        if (A[x*m+y]==9){
                                A[x*m+y]=0;
                        }
                }
        }

        return result;
}
```