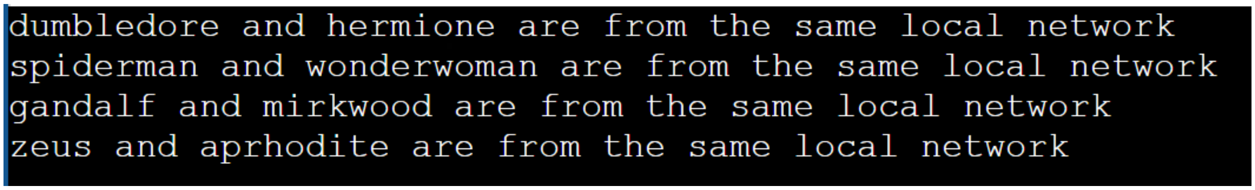
1.

```
#include <stdio.h>


struct address_t
{
  char nickname[150];
  int a, b, c, d;
};


// a and b are same
int localnet(struct address_t ip1, struct address_t ip2){
    if (ip1.a==ip2.a && ip1.b==ip2.b){
        return 1;
    }
    return 0;
}


int main (){
    // initialize array of structure
    struct address_t iplist[300];
    int count = 0;


    // file reading
    FILE *f = fopen ("data.txt", "r");


    // can scan the ip address as a string and then split, or can idividually scan each int and string
    while (!feof (f)){
      fscanf (f, "%d.%d.%d.%d %s", &iplist[count].a, &iplist[count].b, &iplist[count].c, &iplist[count].d, iplist[count].nickname); // no pointer for array needed
```

```
   count++;
  }
  // close
  fclose(f);

  // compare
  for (int i=0; i<count; i++){
    for (int j=i+1;j<count;j++){
      if (localnet(iplist[i], iplist[j])==1){
        printf("%s and %s are from the same local network\n", iplist[i].nickname, iplist[j].nickname);
      }
    }

  }
 return 0;
}
```



```
dumbledore and hermione are from the same local network
spiderman and wonderwoman are from the same local network
gandalf and mirkwood are from the same local network
zeus and aprhodite are from the same local network
```

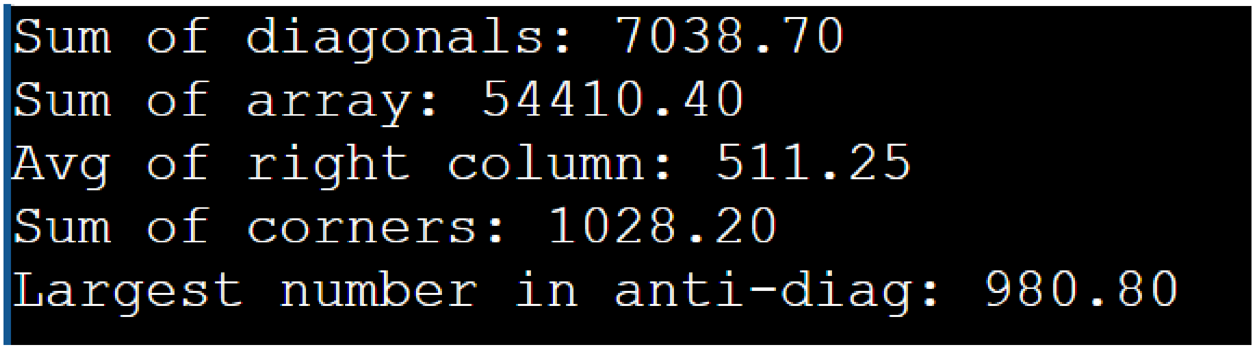
2.

```
#include <stdio.h>
#include "mylibrary.h"
int main()
{
    // file reading
    FILE *f = fopen ("data.txt", "r");

    // array initial
    int ROW=10, r= 0, c=0;
    double a[ROW][ROW];

    // array storage
    while (!feof (f)){
        fscanf(f, "%lf ", &a[r][c]);
        c++;
        if (c > ROW-1){
            c=0;
            r++;
        }
    }

    // close read file
    fclose (f);

    // var saving
    double sumDIAG = sumdiag(a);
    double sumALL = sumall(a);
double avgright = avright(a);

double CORNS = corners(a);

double Lanti = largeanti(a);


// write to b file

FILE *fil = fopen("results.bin", "wb");

fwrite(&sumDIAG, sizeof(sumDIAG), 1, fil);

fwrite(&sumALL, sizeof(sumALL), 1, fil);

fwrite(&avgright, sizeof(avgright), 1, fil);

fwrite(&CORNS, sizeof(CORNS), 1, fil);

fwrite(&Lanti, sizeof(Lanti), 1, fil);

fclose(fil);


// read from b file

FILE *fi = fopen("results.bin", "rb");

double A,B,C,D,E;


while (!feof (fi)){

   fread(&A, sizeof(sumDIAG), 1, fi);

   fread(&B, sizeof(sumALL), 1, fi);

   fread(&C, sizeof(avgright), 1, fi);

   fread(&D, sizeof(CORNS), 1, fi);

   fread(&E, sizeof(Lanti), 1, fi);

}

fclose(fi);


printf("Sum of diagonals: %.2lf\n"

      "Sum of array: %.2lf\n"

      "Avg of right column: %.2lf\n"
```

```
            "Sum of corners: %.2lf\n"
            "Largest number in anti-diag: %.2lf\n", A,B,C,D,E);


    // end


    return 0;
}
```

My lib:

```
//sumdiag: sums all the numbers in the main diagonal of the array ([0][0] to [9][9])
int ROW =10;
double sumdiag(double a[ROW][ROW]){
   double sum=0;
   for (int i=0;i<ROW;i++){
      sum+=a[i][i];
   }
   return sum;
}

//sumall: sums all the numbers in the array.
double sumall(double a[ROW][ROW]){
   double count =0;
   for (int i=0;i<ROW;i++){
      for (int j=0;j<ROW;j++){
         count += a[i][j];
      }
   }
   return count;
}

//avright: calculates the average of the last (rightmost) column of the array.
double avright(double a[ROW][ROW]){
   double avg = 0;
   for (int i=0;i<ROW;i++){
      avg+=a[i][ROW-1];
   }
   return (avg/ROW);
}

//corners: sums the four corners of the array.
double corners(double a[ROW][ROW]){
   return (a[0][0]+a[0][ROW]+a[ROW][0]+a[ROW][ROW]);
}

//largeanti: returns the largest number found in the antidiagonal ([0][9] to [9][0]) of the array.
double largeanti(double a[ROW][ROW]){
   double MAX=a[0][0];

   for (int i=0;i<ROW;i++){
      if (a[i][ROW-1-i]>MAX){
         MAX=a[i][ROW-1-i];
      }
   }
   return MAX;
}
```



```
Sum of diagonals: 7038.70
Sum of array: 54410.40
Avg of right column: 511.25
Sum of corners: 1028.20
Largest number in anti-diag: 980.80
```

pxl