سوال 1 بخش الف :

```
int calculateAndPrintDiagonalSums(int N, int matrix[N][N])
{
    int Diagnal_Sum=0 ,seconed_Diagnal_sum=0 ;
    int n=N-1;
    for(int i=0 ;i<N ;i+=1)
    {
        Diagnal_Sum+= *((int *)matrix + i * N + i);
        seconed_Diagnal_sum+=*((int *)matrix + i * N + n);
        n-=1 ;
    }
    printf("%d\n%d",Diagnal_Sum,seconed_Diagnal_sum);
    return Diagnal_Sum+seconed_Diagnal_sum ;

}
```

سوال 1 بخش ب :

```
bool IsDiagonalMatrix(int N, int matrix[N][N])
{

    for(int i=0 ; i<N ; i++)
    {
        for(int j=0 ; j<N ; j++)
        {
            if (i==j) continue;
            if (*((int *)matrix + i * N + j)!=0) return false ;
        }
    }
    return true ;
}
```

سوال 2 :

خروجی کد 1403 است زیرا ptr یک array pointer است و با عمل ptr+=sizeof int به ایندکس این ارایه اشاره می شود و به طور معمول 4 int بایتی هست یعنی ما به ایندکس شماره 4 یا همان عضو شماره 5 اشاره میکند که عدد 6 هست و بعد در پرانتز عدد منفی 2 که به معنای این هست که هر ایندکسی هست 2 تا به عقب حرکت کند یعنی ایندکس 2 یا همان عضو سوم که عدد 1403 هست و بعد استرینگ 1403 رو به عنوان یک آدرس مساوی با ادرس ptr1 میگذاریم و بعد ادرس ptr1 را به صورت استرینگ که همان 1403 هست پرینت میکنیم

سوال 3 :

```c
void insert(int *addr, int *new_num, int *size)
{
    int index=0 ;
    for(int i=0 ;i<*size ; i++)
    {
        if(*((int *)addr+index)<*new_num)
        {
            index+=1 ;
            continue;
        }
        else break;
    }
    addr = realloc(addr, ((*size) + 1) * sizeof(int));
    int tmp_1=0,tmp_2=0 ;
    for(int i=0 ; i<(*size)+1;i++)
    {
        if(i<index) continue;
        else if (i==index)
        {
            tmp_1=*((int *)addr+index);
            *((int *)addr+index)=*new_num;
        }
        else
        {
            tmp_2=*((int *)addr+i);
            *((int *)addr+i)=tmp_1 ;
            tmp_1=tmp_2 ;
        }
    }
    *size+=1 ;
}
```

سوال 4 الف :

```
//basicly pointer to pointer means a variable that contains another variable(pointer)
address and it is usefull in Dynamic Memory Allocation and Modifying Pointer in Function
// some example :
int x =3 ;
int *ptr=&x ;
int **pptr=&ptr ;
// x is same with *ptr and **pptr
//pptr is same with address of ptr
//ptr is same with address of x
```

سوال 4 ب :

1-از مزایای pass by reference میتوان دسترسی مستقیم به خانه های ممووری دانست که به لطف همین خاصیت ما در توابع میتوانیم مقدار متغیری را عوض کنیم و در تمام کد آن متغیر مقدارش عوض شود 2- از دیگر مزایا میتوان به بهینه بودن و سرعت بالای این روش نام برد منطقی است که دسترسی مستقیم به معنای دسترسی سریع هم هست به طور مثال دسترسی به اعضای یک آرایه به وسیله pass by reference سریع تر از ایندکس دهی هست

1-از معایب pass by reference میتوان به باگ های عظیم در صورت حواس پرتی های کوچک بشود که بعضی مواقع میبینم وقتی برنامه را ران میکنیم اتفاقات عجیب غریبی می افتد به اصطلاح undefined behavior که همین موضوع میتواند به 2-عیب دوم یعنی سخت تر شدن کار دیباگینگ منجر شود (به نحوی دسترسی مستقیم چاقو دولبه است)

سوال 5 امتیازی :

```c
int* mergeAndSortArrays(int *array1, int N1, int *array2, int N2) {
    int *mergedArray = malloc((N1 + N2) * sizeof(int));
    if (mergedArray==NULL) {
        printf("there is no memmory to allocate\n");
        return -1;
    }
    int i = 0, j = 0, k = 0,flag=0;
    while (1) {//sorting by itterating
        if(i>=N1)
        {
            flag=2 ; //making flag for remain  N2
            break;
        }
        if(j>=N2)
        {
            flag=1 ; //making flag for remain  N1
            break;
        }
        if (*(array1+i) <= *(array2+j)) {
            *(mergedArray+k) = *(array1+i);
            i+=1 ;
            k+=1 ;
        } else {
            *(mergedArray+k) = *(array2+j);
            j+=1 ;
            k+=1 ;
        }
    }
    if(flag==1)//use flag to add remain
    {
        while (i < N1) {
            mergedArray[k] = array1[i];
            i+=1;
            k+=1 ;
        }
    }
    else if (flag==2)
    {
        while (j < N2) {
            *(mergedArray+k) = *(array2+j);
            j+=1 ;
            k+=1 ;
        }
    }
    return mergedArray;
}
```