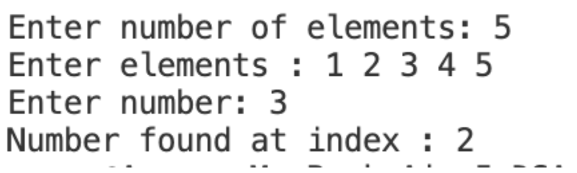
DSA Assignment 2

1.

```
// Binary search for sorted array

#include <iostream>
using namespace std;
int main(){
    int num,n,count=0,i;
    cout<<"Enter number of elements: ";
    cin>>num;
    int a[num],s=0,e=num-1,mid;
    cout<<"Enter elements : ";
    for(i=0;i<num;i++){
        cin>>a[i];
    }
    cout<<"Enter number: ";
    cin>>n;
    while(s<=e){
        mid= s + (e-s)/2;
        if(n==a[mid]){
            count=1;
            break;
        }
        else if(n>a[mid]){
            s=mid+1;
        }
        else{
            e=mid-1;
        }
    }
    if(count==1){
        cout<<"Number found at index : "<<mid<<endl;
    }
    else{
        cout<<"Number not found";
    }

}
```

OUTPUT

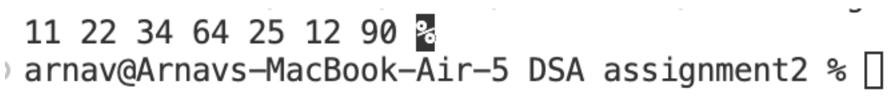


```
Enter number of elements: 5
Enter elements : 1 2 3 4 5
Enter number: 3
Number found at index : 2
```

2.

```
// Bubble sort

# include <iostream>
using namespace std;
int main(){
    int a[7]={64,34,25,12,22,11,90};
    int i,j,temp;
    for(i=0;i<6;i++){
        for(j=i+1;j<7-i-1;j++){
            if(a[i]>a[j]){
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    for(i=0;i<7;i++){
        cout<<a[i]<<' ';
    }
}
```

OUTPUT

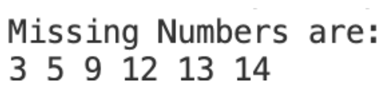


```
11 22 34 64 25 12 90 %
arnav@Arnavs-MacBook-Air-5 DSA assignment2 %
```

3.

```
#include <iostream>
using namespace std;

void findMissing(int arr[], int n) {
    cout << "Missing Numbers are:\n";
    for (int i = 0; i < n - 1; i++) {
        int diff = arr[i+1] - arr[i];
        if (diff > 1) {
            for (int j = arr[i] + 1; j < arr[i+1]; j++) {
                cout << j << " ";
            }
        }
    }
    cout << endl;
}

int main() {
    int arr[] = {1, 2, 4, 6, 7, 8, 10, 11, 15, 16}; //example array
    int n = sizeof(arr) / sizeof(arr[0]);
    findMissing(arr, n);
    return 0;
}
```

OUTPUT

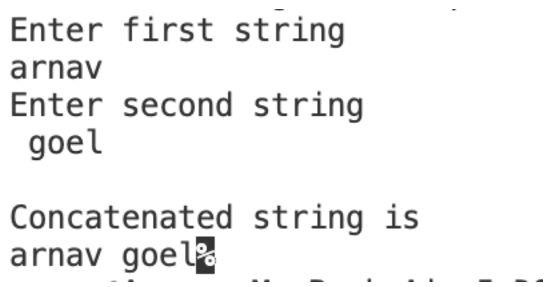


```
Missing Numbers are:
3 5 9 12 13 14
```

4 (a).

```
// Concatenate one string to another

#include <iostream>
#include <string>
using namespace std;
int main(){
    string str1;
    string str2;
    cout<<"Enter first string\n";
    getline(cin,str1);
    cout<<"Enter second string\n";
    getline(cin,str2);
    string str3 = str1 + str2;
    cout<<"\nConcatenated string is \n"<<str3;
}
```

OUTPUT

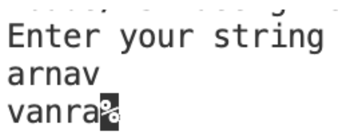


```
Enter first string
arnav
Enter second string
 goel

Concatenated string is
arnav goel%
```

(b).

```
// Reversing of string

#include <iostream>
#include <string>
using namespace std;
int main(){
    string str1;
    int i=0;
    char temp;
    cout<<"Enter your string\n";
    getline(cin,str1);
    int l=str1.length();
    while(i<(l/2)){
        temp=str1[i];
        str1[i]=str1[l-i-1];
        str1[l-i-1]=temp;
        i++;
    }
    cout<<str1;
}
```

OUTPUT

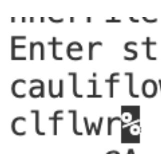


```
Enter your string
arnav
vanra%
```

(c).

```
// Delete all vowels from string

#include <iostream>
#include <string>
using namespace std;
int main(){
    string s;
    int i=0,j;
    cout<<"Enter string\n";
    getline(cin,s);
    int l=s.length();
    while(s[i]!='\0'){
        if(s[i]=='a' || s[i]=='e' || s[i]=='i' || s[i]=='o' || s[i]=='u' ||
s[i]=='A' || s[i]=='E' || s[i]=='I' || s[i]=='O' || s[i]=='U'){
            for(j=i;j<l-1;j++){
                s[j]=s[j+1];
            }
            s[l-1]='\0';
            i--; // This helps us to identify if some vowels are repeated
continuously
        }
        i++;
    }
    cout<<s;
}
```

OUTPUT

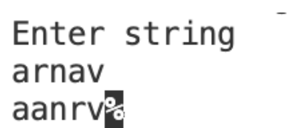


```
Enter string
cauliflower
clflwr
```

(d).

```
// Write a program to sort string in alphabatical order

#include <iostream>
#include <string>
using namespace std;
int main(){
    string s;
    int i=0,j;
    char temp;
    cout<<"Enter string\n";
    getline(cin,s);
    int l=s.length();
    while(s[i]!='\0'){ // converts all letters to lower case
        if(s[i]>=65 & s[i]<=90){
            s[i]+=32;
        }
        i++;
    }
    for(i=0;i<l-1;i++){  // Bubble sort program which sorts according to ascii
code
        for(j=i+1;j<l;j++){
            if(s[i]>s[j]){
                temp=s[i];
                s[i]=s[j];
                s[j]=temp;
            }
        }
    }
    cout<<s;

}
```

OUTPUT

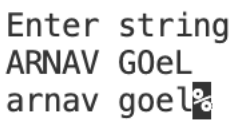


```
Enter string
arnav
aanrv
```

(e).

```cpp
// Conversion of uppercase to lowercase

#include <iostream>
#include <string>
using namespace std;
int main(){
    string s;
    int i=0;
    cout<<"Enter string\n";
    getline(cin,s);
    while(s[i]!='\0'){
        if(s[i]>=65 & s[i]<=90){
            s[i]+=32;
        }
        i++;
    }
    cout<<s;
}
```

OUTPUT

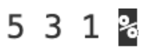


```
Enter string
ARNAV GOeL
arnav goel%
```

5 (a).

```cpp
#include <iostream>
using namespace std;
int main(){
    int a[3][3]={{5,0,0},{0,3,0},{0,0,1}}; // Assume this diagonal matrix
    int b[3]; // In this we will store all the elements of the diagonal matrix
    int i=0,j=0;
    while(i<3 & j<3){
        b[i]=a[i][j];
        i++;
        j++;
    }
    for(i=0;i<3;i++){
        cout<<b[i]<<' ';
    }
}
```

OUTPUT



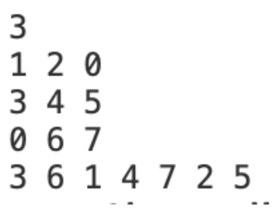
```
5 3 1 %
```

(b).

```cpp
#include <iostream>
using namespace std;
```

```
int main() {
    int n;
    cin >> n;

    int a[n][n], b[(3*n)-2];

    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            cin >> a[i][j];
        }
    }

    int k=0;
    for(int i=1; i<n; i++) {
        b[k++] = a[i][i-1];
    }
    for(int i=0; i<n; i++) {
        b[k++] = a[i][i];
    }
    for(int i=0; i<n-1; i++) {
        b[k++] = a[i][i+1];
    }

    for(int i=0; i<(3*n-2); i++) {
        cout << b[i] << " ";
    }
    cout << endl;

    return 0;
}
```

OUTPUT

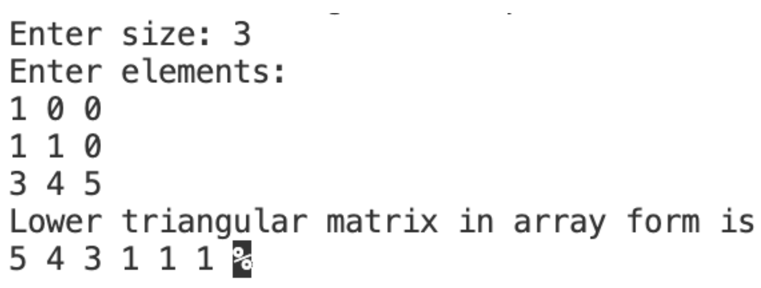


```
3
1 2 0
3 4 5
0 6 7
3 6 1 4 7 2 5
```

(c).

```
// Store lower triangular matrix in array

#include <iostream>
using namespace std;
int main(){
    int n,i,j,k=0;
    cout<<"Enter size: ";
    cin>>n;
    int a[n][n];
    int b[(n*(n+1))/2]; // We will store here
    cout<<"Enter elements: \n";
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cin>>a[i][j];
        }
    }
    i=n-1;
    while(i>=0){
        j=i;
        while(j>=0){
            b[k]=a[i][j];
            k++;
            j--;
        }
        i--;
    }
    cout<<"Lower triangular matrix in array form is \n";
    for(i=0;i<k;i++){
        cout<<b[i]<<' ';
    }
}
```

OUTPUT

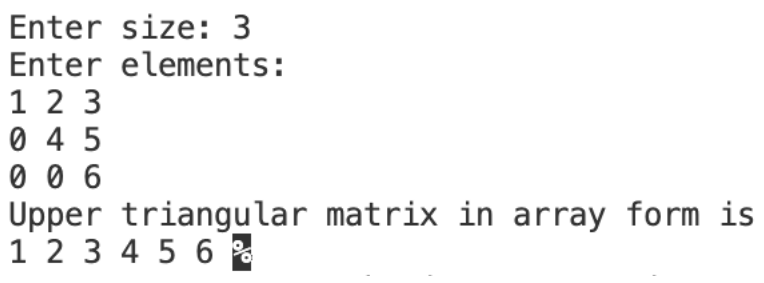


```
Enter size: 3
Enter elements:
1 0 0
1 1 0
3 4 5
Lower triangular matrix in array form is
5 4 3 1 1 1 %
```

(d).

```
// Store uppper triangular matrix in array

#include <iostream>
using namespace std;
int main(){
    int n,i,j,k=0;
    cout<<"Enter size: ";
    cin>>n;
    int a[n][n];
    int b[(n*(n+1))/2]; // We will store here
    cout<<"Enter elements: \n";
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cin>>a[i][j];
        }
    }
    i=0;
    while(i<n){
        j=i;
        while(j<n){
            b[k]=a[i][j];
            k++;
            j++;
        }
        i++;
    }
    cout<<"Upper triangular matrix in array form is \n";
    for(i=0;i<k;i++){
        cout<<b[i]<<' ';
    }
}
```

OUTPUT



```
Enter size: 3
Enter elements:
1 2 3
0 4 5
0 0 6
Upper triangular matrix in array form is
1 2 3 4 5 6 %
```

(e).

```cpp
#include <iostream>
using namespace std;

void printArray(int arr[], int size){
    for(int i = 0; i<size; i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}

int main(){
    int a[3][3]={{1,2,3},{2,4,5},{3,5,6}};
    int b[6];
    int c[3][3];
    int i,j,k=0;

    // store upper triangular part in b
    for(i=0;i<3;i++){
        for(j=i;j<3;j++){
            b[k]=a[i][j];
            k++;
        }
    }
    printArray(b,6);

    // reconstruct matrix c
    k=0;
    for(i=0;i<3;i++){
        for(j=i;j<3;j++){
            c[i][j]=b[k];
            c[j][i]=b[k]; // symmetric property
            k++;
        }
    }

    // print reconstructed matrix
    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            cout<<c[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

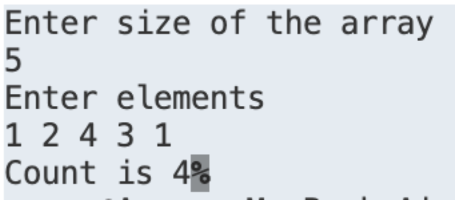
OUTPUT

```
1 2 3 4 5 6
1 2 3
2 4 5
3 5 6
```

7.

```cpp
#include <iostream>
using namespace std;
int main(){
    int size,i,j;
    cout<<"Enter size of the array\n";
    cin>>size;
    int count=0;
    int a[size];
    cout<<"Enter elements\n";
    for(i=0;i<size;i++){
        cin>>a[i];
    }
    for(i=0;i<size-1;i++){
        for(j=i+1;j<size;j++){
            if(a[i]>a[j]){
                count++;
            }
        }
    }
    cout<<"Count is "<<count;
}
```

OUTPUT

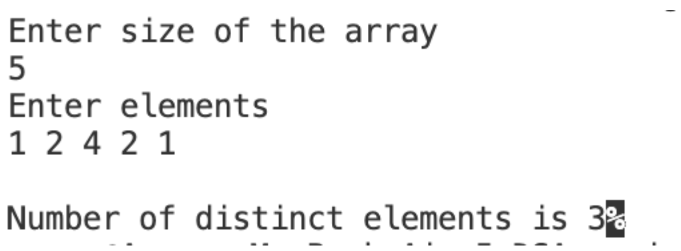


```
Enter size of the array
5
Enter elements
1 2 4 3 1
Count is 4%
```

8.

```cpp
#include <iostream>
using namespace std;

int main() {
    int size;
    cout << "Enter size of the array\n";
    cin >> size;

    int a[size];
    cout << "Enter elements\n";
    for (int i = 0; i < size; i++) {
        cin >> a[i];
    }

    int count = 0; // to count distinct numbers
    for (int i = 0; i < size; i++) {
        bool isDistinct = true;

        // check if a[i] appeared before
        for (int j = 0; j < i; j++) {
            if (a[i] == a[j]) {
                isDistinct = false;
                break;
            }
        }

        if (isDistinct) {
            count++;
        }
    }

    cout << "\nNumber of distinct elements is " << count;
}
```

OUTPUT

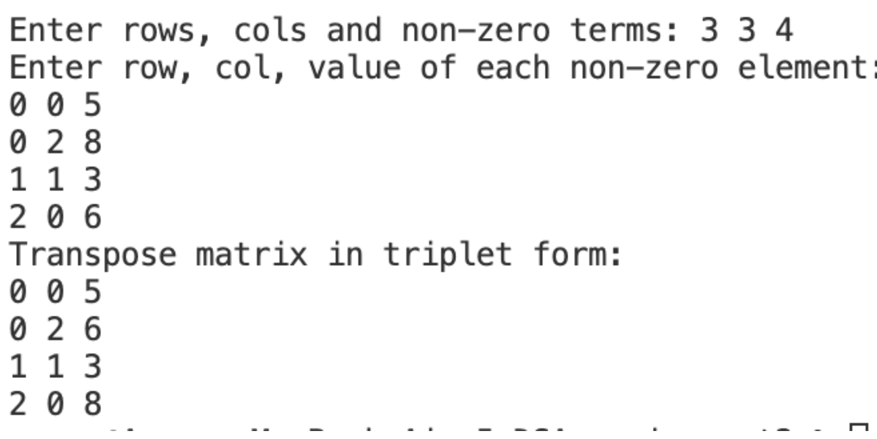


```
Enter size of the array
5
Enter elements
1 2 4 2 1

Number of distinct elements is 3%
```

6(a).

```
#include <iostream>
using namespace std;

struct Term {
    int row, col, val;
};

int main() {
    int rows, cols, terms;
    cout << "Enter rows, cols and non-zero terms: ";
    cin >> rows >> cols >> terms;

    Term a[50], b[50];

    cout << "Enter row, col, value of each non-zero element:\n";
    for (int i = 0; i < terms; i++) {
        cin >> a[i].row >> a[i].col >> a[i].val;
    }

    int k = 0;
    // Transpose -> swap row and col
    for (int i = 0; i < cols; i++) { // scan column-wise
        for (int j = 0; j < terms; j++) {
            if (a[j].col == i) {
                b[k].row = a[j].col;
                b[k].col = a[j].row;
                b[k].val = a[j].val;
                k++;
            }
        }
    }
    cout << "Transpose matrix in triplet form:\n";
    for (int i = 0; i < k; i++) {
        cout << b[i].row << " " << b[i].col << " " << b[i].val << "\n";
    }

    return 0;
}
```

OUTPUT



```
Enter rows, cols and non-zero terms: 3 3 4
Enter row, col, value of each non-zero element:
0 0 5
0 2 8
1 1 3
2 0 6
Transpose matrix in triplet form:
0 0 5
0 2 6
1 1 3
2 0 8
```

(b).

```
#include <iostream>
using namespace std;

struct Term {
    int row, col, val;
};

int main() {
    int rows, cols, termsA, termsB;
    Term A[50], B[50], C[100];

    cout << "Enter rows, cols and non-zero terms for Matrix A: ";
    cin >> rows >> cols >> termsA;
    cout << "Enter row, col, value for Matrix A:\n";
    for (int i = 0; i < termsA; i++) cin >> A[i].row >> A[i].col >> A[i].val;

    cout << "Enter rows, cols and non-zero terms for Matrix B: ";
    cin >> rows >> cols >> termsB;
    cout << "Enter row, col, value for Matrix B:\n";
    for (int i = 0; i < termsB; i++) cin >> B[i].row >> B[i].col >> B[i].val;

    int i = 0, j = 0, k = 0;

    while (i < termsA && j < termsB) {
        if (A[i].row == B[j].row && A[i].col == B[j].col) {
            C[k].row = A[i].row;
            C[k].col = A[i].col;
            C[k].val = A[i].val + B[j].val;
            i++; j++; k++;
        }
        else if (A[i].row < B[j].row || (A[i].row == B[j].row && A[i].col <
B[j].col)) {
            C[k++] = A[i++];
        }
        else {
            C[k++] = B[j++];
        }
    }

    while (i < termsA) C[k++] = A[i++];
    while (j < termsB) C[k++] = B[j++];

    cout << "Sum matrix in triplet form:\n";
    for (int p = 0; p < k; p++) {
        cout << C[p].row << " " << C[p].col << " " << C[p].val << "\n";
    }

    return 0;
}
```

(c)

```
#include <iostream>
using namespace std;

struct Term {
    int row, col, val;
};

int main() {
    int rowsA, colsA, termsA, rowsB, colsB, termsB;
    Term A[50], B[50], C[100];

    cout << "Enter rows, cols and non-zero terms for Matrix A: ";
    cin >> rowsA >> colsA >> termsA;
    cout << "Enter row, col, value for Matrix A:\n";
    for (int i = 0; i < termsA; i++) cin >> A[i].row >> A[i].col >> A[i].val;

    cout << "Enter rows, cols and non-zero terms for Matrix B: ";
    cin >> rowsB >> colsB >> termsB;
    cout << "Enter row, col, value for Matrix B:\n";
    for (int i = 0; i < termsB; i++) cin >> B[i].row >> B[i].col >> B[i].val;

    if (colsA != rowsB) {
        cout << "Multiplication not possible!\n";
        return 0;
    }

    int k = 0;
    // Multiply
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsB; j++) {
            int sum = 0;
            for (int p = 0; p < termsA; p++) {
                if (A[p].row == i) {
                    for (int q = 0; q < termsB; q++) {
                        if (B[q].col == j && B[q].row == A[p].col) {
                            sum += A[p].val * B[q].val;
                        }
                    }
                }
            }
            if (sum != 0) {
                C[k].row = i;
                C[k].col = j;
                C[k].val = sum;
                k++;
            }
        }
    }

    cout << "Product matrix in triplet form:\n";
    for (int p = 0; p < k; p++) {
        cout << C[p].row << " " << C[p].col << " " << C[p].val << "\n";
    }

    return 0;
}
```