AQ1

```cpp
#include <iostream>
using namespace std;
int main()
{
    int n, x;
    cout << "Enter size of array: ";
    cin >> n;
    int arr[n];
    cout << "Enter elements in sorted order: ";
    for (int i = 0; i < n; i++)
    {
        cin >> arr[i];
    }
    cout << "Enter element to search: ";
    cin >> x;
    int l=0,r=n-1,mid,f=-1;
    while (l<=r)
    {
        mid=(l+r)/2;
        if (arr[mid]==x)
        {
            f=mid;
            break;
        }
        else if(arr[mid]<x)
        {
            l=mid+1;
        }
        else
        {
            r=mid-1;
        }
    }
    if (f!=-1)
    {
        cout << "Element found at index: "<<f<<endl;
    }
    else
    {
        cout <<"Element not found"<<endl;
    }
}
```

Output:

```
PS F:\Work\SEM3\DSA\LAB\2> cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ1.cpp -o AQ1 } ; if ($?) { .\AQ1 }
Enter size of array: 5
Enter elements in sorted order: 3
2
1
4
7
Enter element to search: 4
Element found at index: 3
PS F:\Work\SEM3\DSA\LAB\2>
```

AQ2

```cpp
#include <iostream>
using namespace std;
int main()
{
    int arr[]={64, 34, 25, 12, 22, 11, 90};
    int n=7;
    for (int i=0;i<n-1;i++)
    {
        for (int j=0;j<n-i-1;j++)
        {
            if (arr[j]>arr[j+1])
            {
                int t = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] =t;
            }
        }
    }
    cout << "Sorted array: ";
    for (int i=0; i<n;i++)
    {
        cout <<arr[i]<<" ";
    }
    cout << endl;
}
```

Output:

```
PS F:\Work\SEM3\DSA\LAB\2>
                        > cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ2.cpp -o AQ2 } ; if ($?) { .\AQ2 }
Sorted array: 11 12 22 25 34 64 90
PS F:\Work\SEM3\DSA\LAB\2>
```

AQ3:

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

int main() {
    int n;
    cout << "Enter the value of n (range 1 to n): ";
    cin >> n;

    int arr[n-1];
    cout << "Enter " << n-1 << " elements of the array: ";
    for (int i = 0; i < n-1; i++) {
        cin >> arr[i];
    }

    // Linear search
    int missing = -1;
    for (int i = 1; i <= n; i++) {
        bool found = false;
        for (int j = 0; j < n-1; j++) {
            if (arr[j] == i) {
                found = true;
                break;
            }
        }
        if (!found) {
            missing = i;
            break;
        }
    }
    cout << "Missing number: " << missing << endl;

    // Binary search

    int low = 0, high = n - 2;
    int missing2 = -1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == mid + 1) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    missing2 = low + 1;
    cout << "Missing number: " << missing2 << endl;

    return 0;
}
```

Output:

```
                        > cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ3.CPP -o AQ3 } ; if ($?) { .\AQ3 }
Enter the value of n (range 1 to n): 8
Enter 7 elements of the array: 1
2
3
4
6
7
8
Missing number: 5
Missing number: 5
PS F:\Work\SEM3\DSA\LAB\2>
```

AQ4:

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

int main()
{
    cout << "Part (a) \n";
    char s1[200], s2[100];
    cout << "Enter the first string: ";
    cin.getline(s1, 200);
    cout << "Enter the second string: ";
    cin.getline(s2, 100);

    int i=0, j=0;
    while(s1[i] != '\0')
    {
        i++;
    }
    while(s2[j] != '\0')
    {
        s1[i]=s2[j];
        i++;
        j++;
    }
    s1[i]='\0';
    cout << "Concatenated string: " << s1 << endl;

    cout << "Part (b) \n";
    char r[200];
    cout << "Enter a string to reverse: ";
    cin.getline(r, 200);
    int n=0;
    while(r[n] != '\0')
    {
```

```
        n++;
    }
    for(int k=0; k < n / 2; k++)
    {
        char t=r[k];
        r[k]=r[n-k-1];
        r[n-k-1]=t;
    }
    cout << "Reversed string: " << r << endl;

    cout << "Part (c) \n";
    char s[200];
    cout << "Enter a string to remove vowels: ";
    cin.getline(s, 200);
    j=0;
    for(int k=0; s[k] != '\0'; k++)
    {
        char c=s[k];
        if ((c != 'a' && c != 'A' && c != 'e' && c != 'E' && c != 'i' && c != 'I' && c
!= 'o' && c != 'O' && c != 'u' && c != 'U'))
        {
            s[j]=s[k];
            j++;
        }
    }
    s[j]='\0';
    cout << "String without vowels: " << s << endl;

    cout << "Part (d) \n";
    int m;
    cout << "Enter number of strings to sort: ";
    cin >> m;
    cin.ignore();
    char arr[m][100];
    for(int k=0; k < m; k++)
    {
        cout << "Enter string " << k + 1 << ": ";
        cin.getline(arr[k], 100);
    }
    for(i=0;i<m-1;i++)
    {
        for(j=i+1;j<m;j++)
        {
            int k=0;
            while(arr[i][k] != '\0' && arr[j][k] != '\0' && arr[i][k] == arr[j][k])
            {
                k++;
            }
            if (arr[i][k] > arr[j][k])
            {
                char temp[100];
                int p=0;
                while(arr[i][p] != '\0')
                {
```

```
                    temp[p]=arr[i][p];
                    p++;
                }
                temp[p]='\0';
                p=0;
                while(arr[j][p] != '\0')
                {
                    arr[i][p]=arr[j][p];
                    p++;
                }
                arr[i][p]='\0';
                p=0;
                while(temp[p] != '\0')
                {
                    arr[j][p]=temp[p];
                    p++;
                }
                arr[j][p]='\0';
            }
        }
    }

    cout << "Sorted strings:\n";
    for(int i=0; i < m; i++)
    {
        cout << arr[i] << endl;
    }

    cout << "Part (e) \n";
    char str1[200];
    cout << "Enter a string in Uppercase to convert to Lowercase: ";
    cin.getline(str1, 200);
    j=0;
    while(str1[j] != '\0')
    {
        if (str1[j] >= 'A' && str1[j] <= 'Z')
        {
            str1[j]=str1[j] + 32;
        }
        j++;
    }
    cout << "String in Lowercase is: " << str1;
    return 0;
}
```

Output:

```
                    > cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ4.cpp -o AQ4 } ; if ($?) { .\AQ4 }
Part (a)
Enter the first string: HELLO
Enter the second string: WORLD
Concatenated string: HELLOWORLD
Part (b)
Enter a string to reverse: WORLD
Reversed string: DLROW
Part (c)
Enter a string to remove vowels: HELLO WORLD
String without vowels: HLL WRLD
Part (d)
Enter number of strings to sort: 4
Enter string 1: APPLE
Enter string 2: ORANGE
Enter string 3: BALL
Enter string 4: ANGEL
Sorted strings:
ANGEL
APPLE
BALL
ORANGE
Part (e)
Enter a string in Uppercase to convert to Lowercase: HELLO WORLD
String in Lowercase is: hello world
PS F:\Work\SEM3\DSA\LAB\2>
```

AQ5:

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

int main() {
    int n;

    cout << "Enter size of Diagonal Matrix n: ";
    cin >> n;
    int diagMat[n][n];
    cout << "Enter the " << n << "x" << n << " diagonal matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cin >> diagMat[i][j];
        }
    }
    cout << "Input Matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cout << diagMat[i][j] << " ";
        }
```

```
        cout << "\n";
    }
    int diag[n];
    for(int i=0;i<n;i++)
    {
        diag[i] = diagMat[i][i];
    }
    cout << "Linear storage (Diagonal): ";
    for(int i=0;i<n;i++)
    {
        cout << diag[i] << " ";
    }
    cout << "\n\n";

    cout << "Enter size of Tri-diagonal Matrix n: ";
    cin >> n;
    int triMat[n][n];
    cout << "Enter the " << n << "x" << n << " matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cin >> triMat[i][j];
        }
    }
    cout << "Input Matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cout << triMat[i][j] << " ";
        }
        cout << "\n";
    }
    int tri[3*n - 2], k=0;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(i==j || i==j+1 || j==i+1)
            {
                tri[k++] = triMat[i][j];
            }
        }
    }
    cout << "Linear storage (Tri-diagonal): ";
    for(int i=0;i<3*n-2;i++) cout << tri[i] << " ";
    cout << "\n\n";

    cout << "Enter size of Lower Triangular Matrix n: ";
    cin >> n;
    int lowMat[n][n];
    cout << "Enter the " << n << "x" << n << " matrix:\n";
    for(int i=0;i<n;i++)
```

```
    {
        for(int j=0;j<n;j++)
        {
            cin >> lowMat[i][j];
        }
    }
    cout << "Input Matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cout << lowMat[i][j] << " ";
        }
        cout << "\n";
    }
    int lower[n*(n+1)/2];
    k=0;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<=i;j++)
        {
            lower[k++] = lowMat[i][j];
        }
    }
    cout << "Linear storage (Lower Triangular): ";
    for(int i=0;i<n*(n+1)/2;i++)
    {
        cout << lower[i] << " ";
    }
    cout << "\n\n";

    cout << "Enter size of Upper Triangular Matrix n: ";
    cin >> n;
    int upMat[n][n];
    cout << "Enter the " << n << "x" << n << " matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cin >> upMat[i][j];
        }
    }
    cout << "Input Matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cout << upMat[i][j] << " ";
        }
        cout << "\n";
    }
    int upper[n*(n+1)/2];
    k=0;
    for(int i=0;i<n;i++)
```

```
    {
        for(int j=i;j<n;j++)
        {
            upper[k++] = upMat[i][j];
        }
    }
    cout << "Linear storage (Upper Triangular): ";
    for(int i=0;i<n*(n+1)/2;i++)
    {
        cout << upper[i] << " ";
    }
    cout << "\n\n";

    cout << "Enter size of Symmetric Matrix n: ";
    cin >> n;
    int symMat[n][n];
    cout << "Enter the " << n << "x" << n << " matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cin >> symMat[i][j];
        }
    }
    cout << "Input Matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cout << symMat[i][j] << " ";
        }
        cout << "\n";
    }
    int sym[n*(n+1)/2];
    k=0;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<=i;j++)
        {
            sym[k++] = symMat[i][j];
        }
    }
    cout << "Linear storage (Symmetric): ";
    for(int i=0;i<n*(n+1)/2;i++)
    {
        cout << sym[i] << " ";
    }
    cout << "\n";

    return 0;
}
```

Output:

```
PS F:\Work\SEM3\DSA\LAB\2> cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ5.cpp -
o AQ5 } ; if ($?) { .\AQ5 }
Enter size of Diagonal Matrix n: 3
Enter the 3x3 diagonal matrix:
1
0
0
0
2
0
0
0
3
Input Matrix:
1 0 0
0 2 0
0 0 3
Linear storage (Diagonal): 1 2 3

Enter size of Tri-diagonal Matrix n: 4
Enter the 4x4 matrix:
1
3
0
0
3
4
1
0
0
2
3
4
0
0
1
3
Input Matrix:
1 3 0 0
3 4 1 0
0 2 3 4
0 0 1 3
Linear storage (Tri-diagonal): 1 3 3 4 1 2 3 4 1 3

Enter size of Lower Triangular Matrix n: 3
Enter the 3x3 matrix:
1
0
0
5
2
0
6
7
3
Input Matrix:
1 0 0
5 2 0
6 7 3
Linear storage (Lower Triangular): 1 5 2 6 7 3

Enter size of Upper Triangular Matrix n: 3
Enter the 3x3 matrix:
1
4
6
0
2
7
0
0
5
Input Matrix:
1 4 6
0 2 7
0 0 5
Linear storage (Upper Triangular): 1 4 6 2 7 5
```

```
nter size of Symmetric Matrix n: 3
nter the 3x3 matrix:

l

l

nput Matrix:
 1 -1
 2 0
l 0 5
inear storage (Symmetric): 1 1 2 -1 0 5
S F:\Work\SEM3\DSA\LAB\2>
```

AQ6

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

int main()
{
    int r, c;
    int size = 0;
    cout << "Part (a) \n";
    cout << "Enter Row and Column of Original Matrix:" << endl;
    cin >> r >> c;
    int arr[r][c];
    cout << "Enter Matrix elements:" << endl;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            cin >> arr[i][j];
            if ((arr[i][j]) != 0)
            {
                size++;
            }
        }
    }
    int sparse_arr[3][size];
    cout << "Original Matrix is:" << endl;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
    int k = 0;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            if (arr[i][j] != 0)
            {
                sparse_arr[0][k] = i;
                sparse_arr[1][k] = j;
                sparse_arr[2][k] = arr[i][j];
                k++;
            }
        }
    }
    cout << "Sparse Matrix is:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < size; j++)
```

```
        {
            cout << sparse_arr[i][j] << " ";
        }
        cout << endl;
    }
    for (int i = 0; i < size; i++)
    {
        int temp = sparse_arr[0][i];
        sparse_arr[0][i] = sparse_arr[1][i];
        sparse_arr[1][i] = temp;
    }

    cout << "Transpose of Sparse Matrix is:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < size; j++)
        {
            cout << sparse_arr[i][j] << " ";
        }
        cout << endl;
    }

    cout << "Part (b) \n";
    cout << "Enter Row and Column of Matrices:" << endl;
    cin >> r >> c;
    int size_A = 0, size_B = 0;
    int A[r][c];
    int B[r][c];
    cout << "Enter Matrix A elements:" << endl;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            cin >> A[i][j];
            if ((A[i][j]) != 0)
            {
                size_A++;
            }
        }
    }
    int sparse_A[3][size_A];
    cout << "Enter Matrix B elements:" << endl;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            cin >> B[i][j];
            if ((B[i][j]) != 0)
            {
                size_B++;
            }
        }
    }
    int sparse_B[3][size_B];
```

```
    cout << "Original Matrix A is:" << endl;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            cout << A[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Original Matrix B is:" << endl;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            cout << B[i][j] << " ";
        }
        cout << endl;
    }
    k = 0;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            if (A[i][j] != 0)
            {
                sparse_A[0][k] = i;
                sparse_A[1][k] = j;
                sparse_A[2][k] = A[i][j];
                k++;
            }
        }
    }
    cout << "Sparse of Matrix A is:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < size_A; j++)
        {
            cout << sparse_A[i][j] << " ";
        }
        cout << endl;
    }
    k = 0;
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            if (B[i][j] != 0)
            {
                sparse_B[0][k] = i;
                sparse_B[1][k] = j;
                sparse_B[2][k] = B[i][j];
                k++;
            }
        }
```

```
    }
    cout << "Sparse of Matrix B is:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < size_B; j++)
        {
            cout << sparse_B[i][j] << " ";
        }
        cout << endl;
    }
    int size_Sum = 0;
    int Sum[3][size_A + size_B];
    int i = 0, j = 0;
    k = 0;
    while (i < size_A && j < size_B)
    {
        if (sparse_A[0][i] < sparse_B[0][j])
        {
            Sum[0][k] = sparse_A[0][i];
            Sum[1][k] = sparse_A[1][i];
            Sum[2][k] = sparse_A[2][i];
            i++;
            k++;
        }
        else if (sparse_B[0][j] < sparse_A[0][i])
        {
            Sum[0][k] = sparse_B[0][j];
            Sum[1][k] = sparse_B[1][j];
            Sum[2][k] = sparse_B[2][j];
            j++;
            k++;
        }
        else
        {
            if (sparse_A[1][i] < sparse_B[1][j])
            {
                Sum[0][k] = sparse_A[0][i];
                Sum[1][k] = sparse_A[1][i];
                Sum[2][k] = sparse_A[2][i];
                i++;
                k++;
            }
            else if (sparse_B[1][j] < sparse_A[1][i])
            {
                Sum[0][k] = sparse_B[0][j];
                Sum[1][k] = sparse_B[1][j];
                Sum[2][k] = sparse_B[2][j];
                j++;
                k++;
            }
            else
            {
                Sum[0][k] = sparse_A[0][i];
                Sum[1][k] = sparse_A[1][i];
```

```
            Sum[2][k] = sparse_A[2][i] + sparse_B[2][j];
            i++;
            j++;
            k++;
        }
    }
}
while (i < size_A)
{
    Sum[0][k] = sparse_A[0][i];
    Sum[1][k] = sparse_A[1][i];
    Sum[2][k] = sparse_A[2][i];
    i++;
    k++;
}
while (j < size_B)
{
    Sum[0][k] = sparse_B[0][j];
    Sum[1][k] = sparse_B[1][j];
    Sum[2][k] = sparse_B[2][j];
    j++;
    k++;
}
size_Sum = k;
cout << "Addition of Sparse A and B is:" << endl;
for (i = 0; i < 3; i++)
{
    for (j = 0; j < size_Sum; j++)
    {
        cout << Sum[i][j] << " ";
    }
    cout << endl;
}

cout << "Part (c) \n";
int r1, c1, r2, c2;
cout << "Enter Row and Column of Matrix X:" << endl;
cin >> r1 >> c1;
cout << "Enter Row and Column of Matrix Y:" << endl;
cin >> r2 >> c2;
if (c1 == r2)
{
    int X[r1][c1], Y[r2][c2];
    int size_X = 0, size_Y = 0;
    cout << "Enter Matrix X elements:" << endl;
    for (int i = 0; i < r1; i++)
    {
        for (int j = 0; j < c1; j++)
        {
            cin >> X[i][j];
            if ((X[i][j]) != 0)
            {
                size_X++;
            }
```

```
        }
    }
    int sparse_X[3][size_X];
    cout << "Enter Matrix Y elements:" << endl;
    for (int i = 0; i < r2; i++)
    {
        for (int j = 0; j < c2; j++)
        {
            cin >> Y[i][j];
            if ((Y[i][j]) != 0)
            {
                size_Y++;
            }
        }
    }
    int sparse_Y[3][size_Y];
    cout << "Original Matrix X is:" << endl;
    for (int i = 0; i < r1; i++)
    {
        for (int j = 0; j < c1; j++)
        {
            cout << X[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Original Matrix Y is:" << endl;
    for (int i = 0; i < r2; i++)
    {
        for (int j = 0; j < c2; j++)
        {
            cout << Y[i][j] << " ";
        }
        cout << endl;
    }
    k = 0;
    for (int i = 0; i < r1; i++)
    {
        for (int j = 0; j < c1; j++)
        {
            if (X[i][j] != 0)
            {
                sparse_X[0][k] = i;
                sparse_X[1][k] = j;
                sparse_X[2][k] = X[i][j];
                k++;
            }
        }
    }
    cout << "Sparse of Matrix X is:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < size_X; j++)
        {
            cout << sparse_X[i][j] << " ";
```

```
            }
            cout << endl;
        }
        k = 0;
        for (int i = 0; i < r2; i++)
        {
            for (int j = 0; j < c2; j++)
            {
                if (Y[i][j] != 0)
                {
                    sparse_Y[0][k] = i;
                    sparse_Y[1][k] = j;
                    sparse_Y[2][k] = Y[i][j];
                    k++;
                }
            }
        }
        cout << "Sparse of Matrix Y is:" << endl;
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < size_Y; j++)
            {
                cout << sparse_Y[i][j] << " ";
            }
            cout << endl;
        }
        int sparse_P[3][size_X * size_Y];
        int size_P = 0;

        for (int i = 0; i < size_X; i++)
        {
            for (int j = 0; j < size_Y; j++)
            {
                if (sparse_X[1][i] == sparse_Y[0][j])
                {
                    int row = sparse_X[0][i];
                    int col = sparse_Y[1][j];
                    int val = sparse_X[2][i] * sparse_Y[2][j];
                    bool found = false;
                    for (int z = 0; z < size_P; z++)
                    {
                        if (sparse_P[0][z] == row && sparse_P[1][z] == col)
                        {
                            sparse_P[2][z] += val;
                            found = true;
                            break;
                        }
                    }
                    if (!found)
                    {
                        sparse_P[0][size_P] = row;
                        sparse_P[1][size_P] = col;
                        sparse_P[2][size_P] = val;
                        size_P++;
```

```
                    }
                }
            }
        }

        cout << "Sparse of Product is:" << endl;
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < size_P; j++)
                cout << sparse_P[i][j] << " ";
            cout << endl;
        }
    }
    else
    {
        cout << "Multiplication Not Possible" << endl;
    }
    return 0;
}
```

Output:

```
PS F:\Work\SEM3\DSA\LAB\2> cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ6.cpp -o AQ6 } ; if ($?) { .\AQ6 }
Part (a)
Enter Row and Column of Original Matrix:
2
3
Enter Matrix elements:
1
2
3
4
5
6
Original Matrix is:
1 2 3
4 5 6
Sparse Matrix is:
0 0 0 1 1 1
0 1 2 0 1 2
1 2 3 4 5 6
Transpose of Sparse Matrix is:
0 1 2 0 1 2
0 0 0 1 1 1
1 2 3 4 5 6
```

```
Part (b)
Enter Row and Column of Matrices:
3
3
Enter Matrix A elements:
1
6
0
0
0
0
4
0
5
Enter Matrix B elements:
0
0
1
3
0
0
0
7
0
Original Matrix A is:
1 6 0
0 0 0
4 0 5
Original Matrix B is:
0 0 1
3 0 0
0 7 0
Sparse of Matrix A is:
0 0 2 2
0 1 0 2
1 6 4 5
Sparse of Matrix B is:
0 1 2
2 0 1
1 3 7
Addition of Sparse A and B is:
0 0 0 1 2 2 2
0 1 2 0 0 1 2
1 6 1 3 4 7 5
```

```
Part (c)
Enter Row and Column of Matrix X:
3
2
Enter Row and Column of Matrix Y:
2
3
Enter Matrix X elements:
0
9
0
5
1
0
Enter Matrix Y elements:
0
0
3
2
0
0
Original Matrix X is:
0 9
0 5
1 0
Original Matrix Y is:
0 0 3
2 0 0
Sparse of Matrix X is:
0 1 2
1 1 0
9 5 1
Sparse of Matrix Y is:
0 1
2 0
3 2
Sparse of Product is:
0 1 2
0 0 2
18 10 3
PS F:\Work\SEM3\DSA\LAB\2>
```

AQ7

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

int main()
{
    int n;
    cout << "Enter number of elements: ";
    cin >> n;
    int arr[n];
    cout << "Enter array elements: " << endl;
    for (int i=0;i<n;i++)
    {
        cin >>arr[i];
    }
    cout << "The array is: ";
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
    int count=0;
    for (int i=0;i<n-1;i++)
    {
        for (int j=i+1;j<n;j++)
        {
            if (arr[i]>arr[j])
            {
                count++;
            }
        }
    }
    cout << "The number of inversions is: " << count << endl;
    return 0;
}
```

Output:

```
PS F:\Work\SEM3\DSA\LAB\2> cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ7.cpp -o AQ7 } ; if ($?) { .\AQ7 }
Enter number of elements: 5
Enter array elements:
2
4
1
3
5
The array is: 2 4 1 3 5
The number of inversions is: 3
PS F:\Work\SEM3\DSA\LAB\2>
```

AQ8

```cpp
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout << "Enter number of elements: ";
    cin >> n;
    int arr[n];
    cout << "Enter array elements: "<< endl;
    for(int i=0; i < n; i++)
    {
        cin >> arr[i];
    }
    int count=0;
    for(int i=0; i < n; i++)
    {
        int duplicate=0;
        for(int j=0; j < i; j++)
        {
            if (arr[i] == arr[j])
            {
                duplicate=1;
                break;
            }
        }
        if (duplicate == 0)
        {
            count++;
        }
    }
    cout << "Number of distinct elements: " << count << endl;
    return 0;
}
```

Output:

```
PS F:\Work\SEM3\DSA\LAB\2> cd "f:\Work\SEM3\DSA\LAB\2\" ; if ($?) { g++ AQ8.cpp -o AQ8 } ; if ($?) { .\AQ8 }
Enter number of elements: 8
Enter array elements:
1
2
1
3
5
2
1
3
Number of distinct elements: 4
PS F:\Work\SEM3\DSA\LAB\2>
```