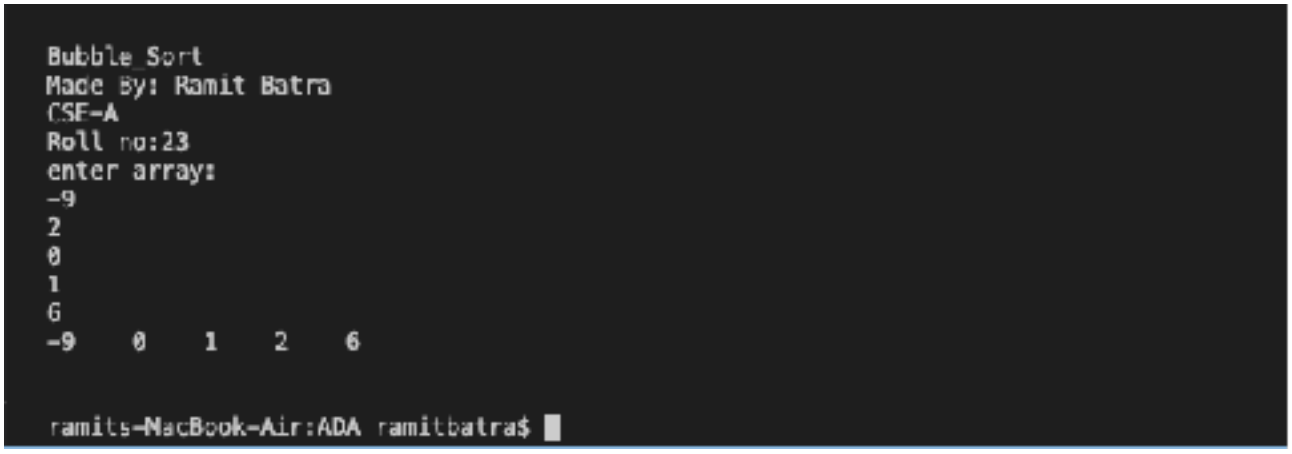
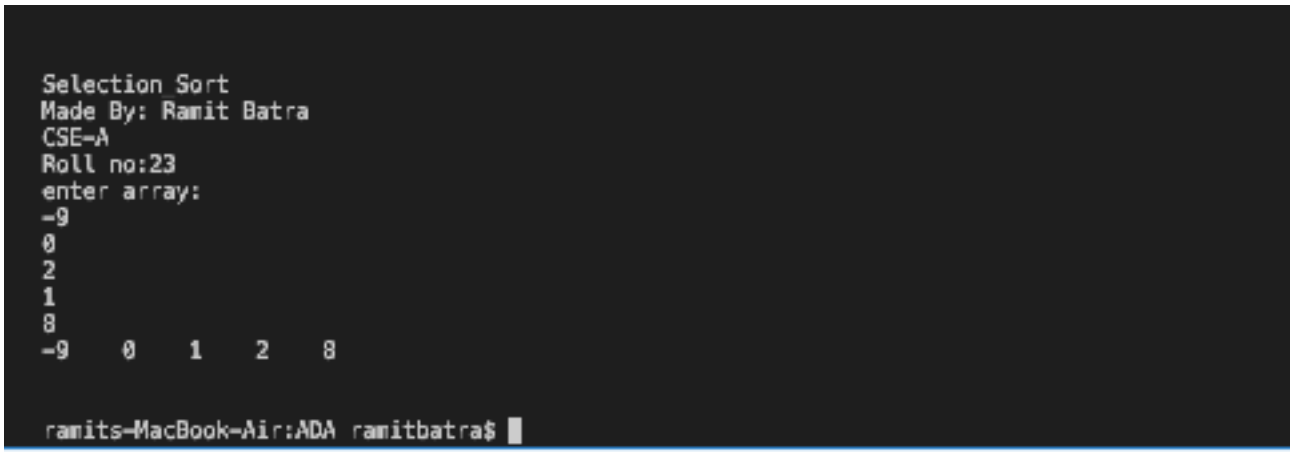
Bubble Sort

```
#include <stdio.h>
#include<iostream>
using namespace std;
int main()
{  int n=5;
    int arr[5];
    cout<<endl<<endl<<endl;
    cout<<"Bubble_Sort\nMade By: Ramit Batra\nCSE-A\nRoll
no:23\nenter array:\n";

    for(int i=0;i<n;i++)
    cin>>arr[i];


for(int j=n-1;j>=0;j--){
     for(int k=0;k<j;k++)
    {
    if(arr[k]>arr[k+1]){

    int temp=arr[k];
    arr[k]=arr[k+1];

    arr[k+1]=temp;}}
}
for(int i=0;i<n;i++)
    cout<<arr[i]<<"     ";
cout<<endl<<endl<<endl;

    return 0;
}
```



Selection Sort

```
#include <stdio.h>
#include<iostream>
using namespace std;
int main()
{
    int n=5,arr[5],min=0;
    cout<<endl<<endl<<endl;
cout<<"Selection_Sort\nMade By: Ramit Batra\nCSE-A\nRoll
no:23\nenter array:\n";
    for(int i=0;i<n;i++)
    cin>>arr[i];
    for(int i=0;i<n;i++){
min=i;
for(int j=i;j<n;j++){
if(arr[j]<arr[min])
min=j;
    }
    int temp=arr[min];
    arr[min]=arr[i];
    arr[i]=temp;
    }
    for(int i=0;i<n;i++)
    cout<<arr[i]<<"    ";
cout<<endl<<endl<<endl;
    return 0;
}
```



Insertion Sort

```
#include <stdio.h>
#include<iostream>
using namespace std;
void insertionSort(int arr[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = arr[i];
        j = i - 1;

        /* Move elements of arr[0..i-1], that are
        greater than key, to one position ahead
        of their current position */
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

// A utility function to print an array of size n
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

/* Driver code */
int main()
{
    int arr[5] ;
    int n = 5;
 cout<<endl<<endl<<endl;
```

```
cout<<"Insertion_Sort\nMade By: Ramit Batra\nCSE-A\nRoll
no:23\nenter array:\n";
for(int i=0;i<n;i++)
    cin>>arr[i];
    insertionSort(arr, n);
    printArray(arr, n);

    return 0;
}
```

Linear Search

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

using namespace std;

class Array {
    public:
        int linearsearch(int arr[], int size, int item) {
            for (int i=0; i<size; i++) {
                if (arr[i]==item) {
                    int pos = i+1;
                    return (pos);
                }
            }
            return (-1);
        }
};

int main() {
    int item, n;
    Array obj;

    cout << "Enter the size of array: ";
    cin >> n;

    int *arr = new int[n];

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

    cout << "Enter the element to be searched: ";
    cin >> item;

    int pos = obj.linearsearch(arr, n, item);
    (pos == -1)
        ? cout << "Element is not present in the array."
```

```
        : cout << "Element "<< item <<" is present at position "
<< pos;

    return 0;
}
```

Binary Search

```
#include <iostream>
#include <stdio.h>
using namespace std;

class Array {
  public:
    int binarysearch(int arr[], int item, int l, int r) {
        if (r >= l) {
            int mid = l + (r - l) / 2;

            // Item in middle
            if (arr[mid] == item)
                return mid;

            // Item in left sub-array
            if (arr[mid] > item)
                return binarysearch(arr, item, l, mid - 1);

            // Item in right sub-array
            return binarysearch(arr, item, mid + 1, r);
        }
        return -1;
    }
};

int main() {
    Array obj;

    int n;
    cout << "Enter the size of array: ";
    cin >> n;
    int *arr = new int[n];
    cout << "Enter " << n*n << " elements in the array: ";

    for (int i=0; i<n; i++)
        cin >> arr[i];

    int item;
    cout << "Enter the element to be searched: ";
```

```
    cin >> item;

    int pos = obj.binarysearch(arr, item, 0, n-1);
    (pos == -1)
        ? cout << "Element is not present in the array."
        : cout << "Element "<< item <<" is present at position "
<< pos;
    return 0;
}
```

Matrix Multiplication

```
#include<stdio.h>
#include<limits.h>
#include<iostream>
int m[10][10]={0},s[10][10]={0},l,k,i,j,q,p[8],n,min,d;

void mat_ch(int);
void optimal(int ,int);

    void mat_ch(int n) {
    for(d=1;d<n;d++) {
        for(i=1;i<=n-d;i++){
            j=i+d;
            min=INT_MAX;
            for(k=i;k<=j-1;k++) {
                q=m[i][k]+m[k+1][j]+ p[i-1] * p[k] * p[j];
                if(q<min) {
                    min=q;
                    s[i][j]=k;
                }
                m[i][j]=min;
            }
        }
    }
}

void optimal(int i,int j) {
    if(i==j) {
        printf("A%d",i);
    }
    else {
        printf("(");
        optimal(i,s[i][j]);
        optimal((s[i][j]+1),j);
        printf(")");
    }
}

int main() {
    printf("enter the matrix size: ");
```

```
    scanf("%d",&n);
    printf("enter the matrix elements: ");
    for(i=0;i<n;i++)
        scanf("%d",&p[i]);
    for(i=1;i<n;i++)
        printf("A%d   %d * %d\n",i,p[i-1],p[i]);
    mat_ch(n);
    printf(" the M matrix\n");
    for(i=0;i<=n;i++) {
        for(j=0;j<=n;j++) {
            if(i>j)
                printf("-\t ");
            else
                printf("%d\t ",m[i][j]);
        }
        printf("\n");
    }
    printf(" the S matrix\n");
    for(i=0;i<=n;i++) {
        for(j=0;j<=n;j++) {
            if(i>j)
                printf("-\t ");
            else
                printf("%d\t ",s[i][j]);
        }
        printf("\n");
    }
    optimal(1,n);
    printf(" \n Minimum Product = %d",m[1][n]);
    //  getch();
    return 0;
}
```

LCS

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

void lcsAlgo(char *S1, char *S2, int m, int n) {
  int LCS_table[m + 1][n + 1];


  // Building the mtrix in bottom-up way
  for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
      if (i == 0 || j == 0)
        LCS_table[i][j] = 0;
      else if (S1[i - 1] == S2[j - 1])
        LCS_table[i][j] = LCS_table[i - 1][j - 1] + 1;
      else
        LCS_table[i][j] = max(LCS_table[i - 1][j], LCS_table[i][j
- 1]);
    }
  }

  int index = LCS_table[m][n];
  char lcsAlgo[index + 1];
  lcsAlgo[index] = '\0';

  int i = m, j = n;
  while (i > 0 && j > 0) {
    if (S1[i - 1] == S2[j - 1]) {
      lcsAlgo[index - 1] = S1[i - 1];
      i--;
      j--;
      index--;
    }

    else if (LCS_table[i - 1][j] > LCS_table[i][j - 1])
      i--;
    else
      j--;
  }
```

```
  // Printing the sub sequences
  cout << "S1 : " << S1 << "\nS2 : " << S2 << "\nLCS: " << lcsAlgo
<< "\n";
}

int main() {
  char S1[] = "ACADBDA";
  char S2[] = "CBDA";
  int m = strlen(S1);
  int n = strlen(S2);

  lcsAlgo(S1, S2, m, n);
}
```