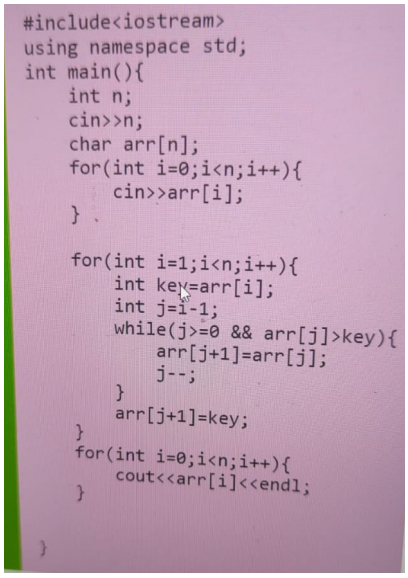
Insertion sort



```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    char arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }

    for(int i=1;i<n;i++){
        int key=arr[i];
        int j=i-1;
        while(j>=0 && arr[j]>key){
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=key;
    }
    for(int i=0;i<n;i++){
        cout<<arr[i]<<endl;
    }

}
```

Negative and positive

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

void ins(int arr[], int n, bool check) {
    for (int i = 1; i < n; i++) {
        int k = arr[i];
        int j = i - 1;
        if (check) {
            while (j >= 0 && arr[j] < k) {
                arr[j + 1] = arr[j];
                j--;
            }
        } else {
            while (j >= 0 && arr[j] > k) {
                arr[j + 1] = arr[j];
                j--;
            }
        }
        arr[j + 1] = k;
    }
}

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

    int arr[n], neg[n], pos[n];
    int nc = 0, pc = 0;
    bool z = false;

    for (int i = 0; i < n; i++) {
        cin >> arr[i];
        if (arr[i] < 0) {
            neg[nc++] = arr[i];
        } else if (arr[i] > 0) {
            pos[pc++] = arr[i];
```

```
        } else {
            z = true;
        }
    }

    // Sort negatives in descending order
    ins(neg, nc, true);

    // Sort positives in ascending order
    ins(pos, pc, false);

    int ix = 0;

    // Add negatives first
    for (int i = 0; i < nc; i++) {
        arr[ix++] = neg[i];
    }

    // Add zero if it exists
    if (z) {
        arr[ix++] = 0;
    }

    // Add positives
    for (int i = 0; i < pc; i++) {
        arr[ix++] = pos[i];
    }

    // Print the sorted array
    for (int i = 0; i < n; i++) {
        cout << arr[i] << endl;
    }

    return 0;
}
```

Coordinates

```
#include <iostream>
#include <cmath>
using namespace std;
double dist(int x1,int y1,int x2,int y2) {
    return sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
}
void ins(int points[][2],int n,int x,int y) {
    for (int i=1;i<n;++i) {
        int X=points[i][0];
        int Y=points[i][1];
        int j=i-1;
        double d=dist(x,y,X,Y);
        while (j>=0&& dist(x,y,points[j][0],points[j][1])>d) {
            points[j+1][0]=points[j][0];
            points[j+1][1]=points[j][1];
            j--;
        }
        points[j+1][0]=X;
        points[j+1][1]=Y;
    }
}
```

```
int main() {
    int n;
    cin>>n;
    int points[n][2];
    for (int i=0;i<n;++i) {
        cin>>points[i][0]>>points[i][1];
    }
    int x=points[0][0];
    int y=points[0][1];
    for (int i=1;i<n;++i) {
        if (points[i][0]<x||(points[i][0]==x&&points[i][1]<y)){
            x=points[i][0];
            y=points[i][1];
        }
    }
    ins(points,n,x,y);
    for (int i=0;i<n;++i) {
        cout<<points[i][0]<<endl;
        cout<<points[i][1]<<endl
    }
    return 0;
}
```

Coordinates 3

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

// Updated function to calculate distance between two points in 3D
double dist(int x1, int y1, int z1, int x2, int y2, int z2) {
    return sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) + (z2-z1)*(z2-z1));
}

// Updated insertion sort function for 3D points
void ins(int points[][3], int n, int x, int y, int z) {
    for (int i = 1; i < n; ++i) {
        int X = points[i][0];
        int Y = points[i][1];
        int Z = points[i][2];
        int j = i - 1;
        double d = dist(x, y, z, X, Y, Z);
        while (j >= 0 && dist(x, y, z, points[j][0], points[j][1], points[j][2]) > d) {
            points[j + 1][0] = points[j][0];
            points[j + 1][1] = points[j][1];
            points[j + 1][2] = points[j][2];
            j--;
        }
        points[j + 1][0] = X;
        points[j + 1][1] = Y;
        points[j + 1][2] = Z;
    }
}

int main() {
    int n;
    cin >> n;
    int points[n][3];  // Updated to store 3 coordinates per point (x, y, z)
    for (int i = 0; i < n; ++i) {
        cin >> points[i][0] >> points[i][1] >> points[i][2];
```

```
    }

    int x = points[0][0];
    int y = points[0][1];
    int z = points[0][2];

    // Updated logic to find the reference point in 3D
    for (int i = 1; i < n; ++i) {
        if (points[i][0] < x ||
            (points[i][0] == x && points[i][1] < y) ||
            (points[i][0] == x && points[i][1] == y && points[i][2] < z)) {
            x = points[i][0];
            y = points[i][1];
            z = points[i][2];
        }
    }

    // Sort the points based on distance from the reference point (x, y, z)
    ins(points, n, x, y, z);

    // Output the sorted points
    for (int i = 0; i < n; ++i) {
        cout << points[i][0] << endl;
        cout << points[i][1] << endl;
        cout << points[i][2] << endl;
    }

    return 0;
}
```

merge sort

Without sentinels

```
#include <iostream>
using namespace std;
void merge(float arr[],int l,int m,int r) {
    int i=l;
    int j=m+1;
    int k=0;
    float temp[r-l+1];
    while (i<=m&&j<=r) {
        if (arr[i]<=arr[j]) {
            temp[k]=arr[i];
            i++;
        } else {
            temp[k]=arr[j];
            j++;
        }
        k++;
    }
    while (i<=m) {
        temp[k]=arr[i];
        i++;
        k++;
    }
    while (j<=r) {
        temp[k]=arr[j];
        j++;
        k++;
```

```
    }
    for (int p=0;p<k;p++) {
        arr[l+p]=temp[p];
    }
}

void merges(float arr[],int l,int r) {
    if (l<r) {
        int m=(l+r)/2;
        merges(arr,l,m);
        merges(arr,m+1,r);
        merge(arr,l,m,r);
    }
}
int main() {
    int n=0;
    cin>>n;
    float arr[n];
    for (int i=0;i<n;i++) {
        cin>>arr[i];
    }
    int l=0;
    int r=n-1;
    merges(arr,l,r)
    for (int i=0;i<n;i++) {
        cout<<arr[i]<<endl;
    }
    return 0;
}
```

Insertion and merge

```
#include <iostream>
using namespace std;
void merge(float arr[],int l,int m,int r) {
    int i=l;
    int j=m+1;
    int k=0;
    float temp[r-l+1];
    while (i<=m&&j<=r) {
        if (arr[i]<=arr[j]) {
            temp[k]=arr[i];
            i++;
        } else {
            temp[k]=arr[j];
            j++;
        }
        k++;
    }
    while (i<=m) {
        temp[k]=arr[i];
        i++;
        k++;
    }
    while (j<=r) {
        temp[k]=arr[j];
        j++;
        k++;
    }
```

```
    for (int p=0;p<k;p++) {
        arr[l+p]=temp[p];
    }
}
void merges(float arr[],int l,int r) {
    if (l<r) {
        int m=(l+r)/2;
        merges(arr,l,m);
        merges(arr,m+1,r);
        merge(arr,l,m,r);
    }
}
void ins(float arr[],int m,int r) {
    for (int i=m+1;i<=r;i++) {
        float tmp=arr[i];
        int j=i-1;
        while (j>=m&&arr[j]>tmp) {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=tmp;
    }
}
int main() {
    int n=0;
    cin>>n;
    float arr[n];
    for (int i=0;i<n;i++) {
        cin>>arr[i];
    }
    int l=0;
    int r=n-1;
    int m=(l+r)/2;
    merges(arr,l,m);
    ins(arr,m+1,r);
    merge(arr,l,m,r);
    for (int i=0;i<n;i++) {
        cout<<arr[i]<<endl;
    }
    return 0;
}
```

subarray<3

```
#include <iostream>
using namespace std;
void merge(float arr[],int l,int m,int r) {
    int i=l;
    int j=m+1;
    int k=0;
    float temp[r-l+1];
    while (i<=m&&j<=r) {
        if (arr[i]<=arr[j]) {
            temp[k]=arr[i];
            i++;
        } else {
            temp[k]=arr[j];
            j++;
        }
        k++;
```

```
    }
    while (i<=m) {
        temp[k]=arr[i];
        i++;
        k++;
    }
    while (j<=r) {
        temp[k]=arr[j];
        j++;
        k++;
    }
    for (int p=0;p<k;p++) {
        arr[l+p]=temp[p];
    }
}
void ins(float arr[],int l,int r) {
    for (int i=l+1;i<=r;i++) {
        float tmp=arr[i];
        int j=i-1;
        while (j>=l&&arr[j]>tmp) {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=tmp;
    }
}
void merges(float arr[],int l,int r) {
    if(r-l+1<=4){
        ins(arr,l,r);
    }
    else{
        int m=(l+r)/2;
        merges(arr,l,m);
        merges(arr,m+1,r);
        merge(arr,l,m,r);
    }
}
int main() {
    int n=0;
    cin>>n;
    float arr[n];
    for (int i=0;i<n;i++) {
        cin>>arr[i];
    }
    merges(arr,0,n-1);
    for (int i=0;i<n;i++) {
        cout<<arr[i]<<endl;
    }
    return 0;
}
```

3 subarrays

```
#include <iostream>
using namespace std;
void merge(float arr[],int l,int m1,int m2,int r) {
    int i=l;
    int j=m1+1;
```

```
    int k=m2+1;
    int n=0;
    float temp[r-l+1];
    while ((i<=m1)&&(j<=m2)&&(k<=r)) {
        if (arr[i]<=arr[j]&&arr[i]<=arr[k]) {
            temp[n++]=arr[i++];
        }
        else if ((arr[j]<=arr[i])&&(arr[j]<=arr[k])) {
            temp[n++]=arr[j++];
        }
        else {
            temp[n++]=arr[k++];
        }
    }
    while (i<=m1&&j<=m2) {
        if (arr[i]<=arr[j]) {
            temp[n++]=arr[i++];
        }
        else {
            temp[n++]=arr[j++];
        }
    }
    while (j<=m2&&k<=r) {
        if (arr[j]<=arr[k]) {
            temp[n++]=arr[j++];
        }
        else {
            temp[n++]=arr[k++];
        }
    }
    while (i<=m1&&k<=r) {
        if (arr[i]<=arr[k]) {
            temp[n++]=arr[i++];
        }
        else {
            temp[n++]=arr[k++];
        }
    }
    while (i<=m1) {
        temp[n++]=arr[i++];
    }
    while (j<=m2) {
        temp[n++]=arr[j++];
    }
    while (k<=r) {
        temp[n++]=arr[k++];
    }
    for (i=l;i<=r;i++) {
        arr[i]=temp[i-l];
    }
}
void merges(float arr[],int l,int r) {
```

```
    if (l<r) {
        int m1=(l+(r-l)/3);
        int m2 =(l+2*(r-l)/3);
        merges(arr,l,m1);
        merges(arr,m1+1,m2);
        merges(arr,m2+1,r);
        merge(arr,l,m1,m2,r);
    }
}
int main() {
    int n;
    cin>>n;
    float arr[n];
    for (int i=0;i<n;i++) {
        cin>>arr[i];
    }
    merges(arr,0,n-1);
    for (int i=0;i<n;i++) {
        cout<<arr[i]<<endl;
    }
    return 0;
}
```

N sub arrays

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

// Function to merge k sorted subarrays into a single sorted array
void merge(float arr[], int l, const vector<int>& mids, int r, int k) {
    vector<int> idx(k + 1);  // Current index in each part
    for (int i = 0; i < k; ++i) {
        idx[i] = (i == 0) ? l : mids[i - 1] + 1;
    }
    idx[k] = r + 1;

    vector<float> temp(r - l + 1);
    int n = 0;

    while (true) {
        int minIdx = -1;
        for (int i = 0; i < k; ++i) {
            if (idx[i] < idx[i + 1] && (minIdx == -1 || arr[idx[i]] < arr[idx[minIdx]])) {
                minIdx = i;
            }
        }
        if (minIdx == -1) break;
        temp[n++] = arr[idx[minIdx]++];
    }
```

```
    for (int i = l; i <= r; ++i) {
        arr[i] = temp[i - l];
    }
}

// Function to perform k-way merge sort
void merges(float arr[], int l, int r, int k) {
    if (l < r) {
        vector<int> mids(k - 1);
        for (int i = 0; i < k - 1; ++i) {
            mids[i] = l + (i + 1) * (r - l) / k;
        }
        for (int i = 0; i < k; ++i) {
            int start = (i == 0) ? l : mids[i - 1] + 1;
            int end = (i == k - 1) ? r : mids[i];
            merges(arr, start, end, k);
        }
        merge(arr, l, mids, r, k);
    }
}

int main() {
    int n, k;
    cout << "Enter number of elements: ";
    cin >> n;
    cout << "Enter the number of divisions (k): ";
    cin >> k;

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

    merges(arr, 0, n - 1, k);

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

    return 0;
}
```

Pivot

```
#include <iostream>
using namespace std;
void merge(float arr[],int l,int m,int r) {
    int i=l;
    int j=m+1;
```

```
    int k=0;
    float temp[r-l+1];
    while (i<=m&&j<=r){
        if (arr[i]<=arr[j]) {
            temp[k]=arr[i];
            i++;
        } else {
            temp[k]=arr[j];
            j++;
        }
        k++;
    }
    while (i<=m) {
        temp[k]=arr[i];
        i++;
        k++;
    }
    while (j<=r) {
        temp[k]=arr[j];
        j++;
        k++;
    }
    for (int p=0;p<k;p++) {
        arr[l+p]=temp[p];
    }
}
void merges(float arr[],int l,int r) {
    if (l<r) {
        for (int i=l+1;i<=r;i++) {
            float pt=arr[i];
            int j=i-1;
            while (j>=l&&arr[j]>pt) {
                arr[j+1]=arr[j];
                j--;
            }
            arr[j+1]=pt;
            merges(arr,l,j);
            merges(arr,j+2,r);
            merge(arr,l,j+1,r);
        }
    }
}

int main() {
    int n;
    cin >> n;
    float arr[n];
    for (int i=0;i<n;i++) {
        cin>>arr[i];
    }
    int l=0;
    int r=n-1;
```

```
    merges(arr,l,r);
    for (int i=0;i<n;i++) {
        cout<<arr[i]<<endl;
    }
    return 0;
}
```

1) REGULAR CASE KADANE'S ALGO O(N)

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

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

   int maxs=INT_MIN;
   int sum=0;
   start=0;
   end=0;
   tempstart=0;

   for(int i=0;i<n;i++){
      sum+=arr[i];
      if(sum>maxs){
         maxs=sum;
         start=tempstart;
         end=i
      }
      if(sum<0){
         sum=0;
         tempstart++;
      }
   }
   cout<<start<<endmaxs;
}
```

NOTE:
VALUE->NO OF ELEMENTS IN SUBARRAY
SUBARRAY->SUM

2) **Maximum Sum with Minimum Number of Elements**

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

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

  int maxs = INT_MIN;
  int sum = 0;
  int start = 0;
  int end = 0;
  int tempstart = 0;

  for(int i = 0; i < n; i++){
     sum += arr[i];

     if(sum > maxs || (sum == maxs && (i - tempstart) < (end - start))){
        maxs = sum;
        start = tempstart;
        end = i;
     }

     if(sum < 0){
        sum = 0;
        tempstart = i + 1;
     }
  }

  cout << "Start: " << start << " End: " << end << " Max Sum: " << maxs << endl;
}
```

3)Maximum Sum with Maximum Number of Elements

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

int main(){
```

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

  int maxs = INT_MIN;
  int sum = 0;
  int start = 0;
  int end = 0;
  int tempstart = 0;

  for(int i = 0; i < n; i++){
     sum += arr[i];

     if(sum > maxs || (sum == maxs && (i - tempstart) > (end - start))){
        maxs = sum;
        start = tempstart;
        end = i;
     }

     if(sum < 0){
        sum = 0;
        tempstart = i + 1;
     }
  }

  cout << "Start: " << start << " End: " << end << " Max Sum: " << maxs << endl;
}
```

4)Minimum Sum with Minimum Number of Elements

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

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

  int mins = INT_MAX;
```

```
  int sum = 0;
  int start = 0;
  int end = 0;
  int tempstart = 0;

  for(int i = 0; i < n; i++){
     sum += arr[i];

     if(sum < mins || (sum == mins && (i - tempstart) < (end - start))){
        mins = sum;
        start = tempstart;
        end = i;
     }

     if(sum > 0){
        sum = 0;
        tempstart = i + 1;
     }
  }

  cout << "Start: " << start << " End: " << end << " Min Sum: " << mins << endl;
}
```

5)Minimum Sum with Maximum Number of Elements

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

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

  int mins = INT_MAX;
  int sum = 0;
  int start = 0;
  int end = 0;
  int tempstart = 0;

  for(int i = 0; i < n; i++){
     sum += arr[i];
```

```
      if(sum < mins || (sum == mins && (i - tempstart) > (end - start))){
         mins = sum;
         start = tempstart;
         end = i;
      }

      if(sum > 0){
         sum = 0;
         tempstart = i + 1;
      }
   }

   cout << "Start: " << start << " End: " << end << " Min Sum: " << mins << endl;
}
```

Normal dcc

```
#include<iostream>
#include<limits.h>
using namespace std;
struct val{
   int low;
   int high;
   int sum;
};
val cross(int arr[],int l,int m,int h){
   int lsum=INT_MIN;
   int rsum=INT_MIN;
   int li=0,ri=0;
   val d;
   int sum=0;
   for(int i=m;i>=l;i--){
      sum+=arr[i];
      if(sum>lsum){
         lsum=sum;
         li=i;
      }
   }
   sum=0;
   for(int i=m+1;i<=h;i++){
      sum+=arr[i];
      if(sum>rsum){
         rsum=sum;
         ri=i;
      }
   }
   d.low=li;
   d.high=ri;
   d.sum=lsum+rsum;
   return d;
```

```
}
val maxsum(int arr[],int l,int h){
    if(l==h){
        return {l,h,arr[l]};
    }
    int m=(l+h)/2;
    val ls=maxsum(arr,l,m);
    val rs=maxsum(arr,m+1,h);
    val cs=cross(arr,l,m,h);
    if(ls.sum>=rs.sum && ls.sum>=cs.sum)
    return ls;
    else if(rs.sum>=ls.sum && rs.sum>=cs.sum)
    return rs;
    else
    return cs;

}

int main() {
    int n;
    cin>>n;
    int arr[n];
    for (int i=0;i<n;i++) {
        cin>>arr[i];
    }
    val ans=maxsum(arr,0,n-1);
    cout<<ans.low+1<<endl;
    cout<<ans.high+1<<endl;
    cout<<ans.sum<<endl;
}
```

Max sum MIn length of sub array

```
#include <climits>
#include <iostream>
using namespace std;
void sub(int arr[], int n) {
    int sum=0;
    int st=0,end=0;
    int mx=INT_MIN;
    int ts=0;
    for (int i=0;i<n;i++) {
        if (arr[i]>=0) {
            sum+=arr[i];
            if (sum>mx) {
                mx=sum;
                st=ts;
                end=i;
            }
        }
```

```
        else {
            sum=0;
            ts=i+1;
        }
    }
    cout<<st+1<<endl;
    cout<<end+1<<endl;
    cout<<mx<<endl;
}
int main() {
    int n;
    cin>>n;
    int arr[n];
    for (int i=0;i<n;i++) {
        cin>>arr[i];
    }
    sub(arr,n);
    return 0;
}
```

Max sum Max length of sub array

```
#include <iostream>
#include <climits>

using namespace std;

void findMaxSumSubarray(int arr[], int n) {//-2,-5,6,-2,-3,1,5,-6
    int max_sum = 0;
    int current_sum = 0;
    int start_index = 0, end_index = 0, temp_start = 0;
    int max_length = 0;

    for (int i = 0; i < n; i++) {
        if (arr[i] >= 0) {
            current_sum += arr[i];
            if (current_sum > max_sum || (current_sum == max_sum && (i - temp_start + 1) >
max_length)) {
                max_sum = current_sum;
                max_length = i - temp_start + 1;
                start_index = temp_start;
                end_index = i;
            }
        } else {
            current_sum = 0;
            temp_start = i + 1;
        }
    }

    if (max_sum > 0) {
        cout << start_index + 1 << endl;
```

```
        cout << end_index + 1 << endl;
        cout << max_sum << endl;
    }
}

int main() {
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    findMaxSumSubarray(arr, n);
    return 0;
}

#include <iostream>
#include <string>

int main() {
    int n;
    std::cout << "Enter the number of words: ";
    std::cin >> n;

    std::string words[n];
    std::cout << "Enter " << n << " words:" << std::endl;

    for (int i = 0; i < n; ++i) {
        std::cin >> words[i];
    }

    std::cout << "You entered:" << std::endl;
    for (int i = 0; i < n; ++i) {
        std::cout << words[i] << std::endl;
    }

    return 0;
}
```