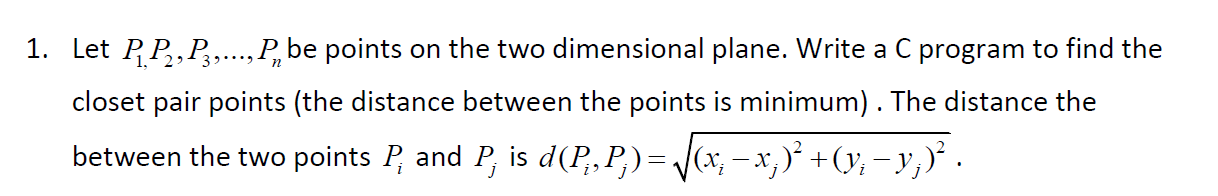
*Name:Ayan Sadhukhan*

*Regd No: 19BEC1448*

*Question 1:*



*Code:*

#include <stdio.h>

#include <float.h>

#include <stdlib.h>

#include <math.h>

struct Point

{

int x, y;

};

int compareX(const void\* a, const void\* b)

{

Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;

return (p1->x - p2->x);

}

int compareY(const void\* a, const void\* b)

{

Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;

return (p1->y - p2->y);

}

float dist(Point p1, Point p2)

{

return sqrt( (p1.x - p2.x)\*(p1.x - p2.x) +

(p1.y - p2.y)\*(p1.y - p2.y)

);

}

float bruteForce(Point P[], int n)

{

float min = FLT\_MAX;

for (int i = 0; i < n; ++i)

for (int j = i+1; j < n; ++j)

if (dist(P[i], P[j]) < min)

min = dist(P[i], P[j]);

return min;

}

float min(float x, float y)

{

return (x < y)? x : y;

}

float stripClosest(Point strip[], int size, float d)

{

float min = d;

qsort(strip, size, sizeof(Point), compareY);

for (int i = 0; i < size; ++i)

for (int j = i+1; j < size && (strip[j].y - strip[i].y) < min; ++j)

if (dist(strip[i],strip[j]) < min)

min = dist(strip[i], strip[j]);

return min;

}

float closestUtil(Point P[], int n)

{

if (n <= 3)

return bruteForce(P, n);

int mid = n/2;

Point midPoint = P[mid];

float dl = closestUtil(P, mid);

float dr = closestUtil(P + mid, n-mid);

float d = min(dl, dr);

Point strip[n];

int j = 0;

for (int i = 0; i < n; i++)

if (abs(P[i].x - midPoint.x) < d)

strip[j] = P[i], j++;

return min(d, stripClosest(strip, j, d) );

}

float closest(Point P[], int n)

{

qsort(P, n, sizeof(Point), compareX);

return closestUtil(P, n);

}

int main()

{

Point P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};

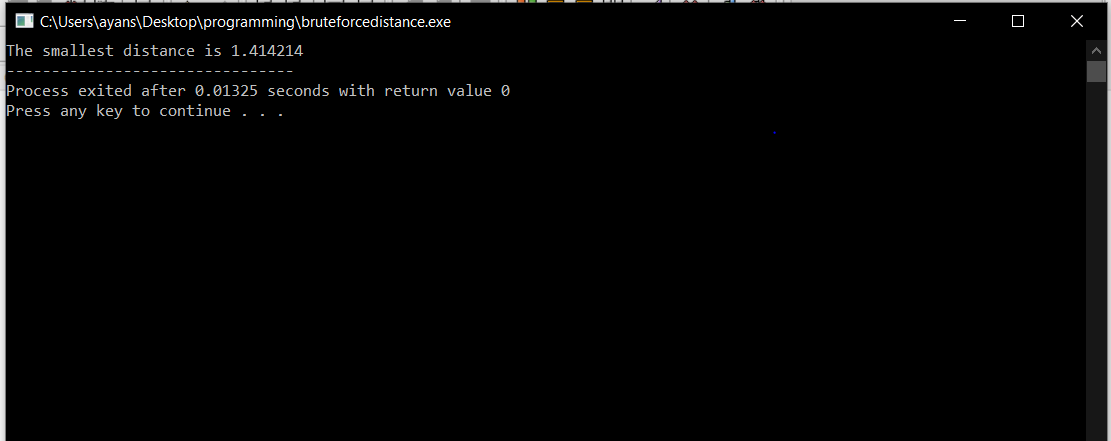
int n = sizeof(P) / sizeof(P[0]);

printf("The smallest distance is %f ", closest(P, n));

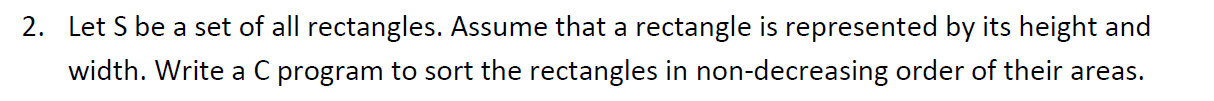
return 0;

}

*Output:*



*Question 2:*



*Code:*

#include<iostream>

#include<string>

using namespace std;

struct triangle

{

int h;

int w;

};

void merge(int arr[], int p, int q, int r)

{

int i, j, k;

int n1 = q - p + 1;

int n2 = r - q;

int L[n1], R[n2];

for (i = 0; i < n1; i++)

L[i] = arr[p + i];

for (j = 0; j < n2; j++)

R[j] = arr[q + 1+ j];

i = 0;

j = 0;

k = p;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int p, int r)

{

if (p < r)

{

int q = (r+p)/2;

mergeSort(arr, p, q);

mergeSort(arr, q+1, r);

merge(arr, p, q, r);

}

}

void printArray(int A[], int size)

{

int i;

for (i=0; i < size; i++)

printf("%d ", A[i]);

printf("\n");

}

int main()

{

int n;

cout<<"Enter the number of triangles :";

cin>>n;

triangle t[n];

for(int i=0;i<n;i++)

{

cout<<"Enter the triangle height :";

cin>>t[i].h;

cout<<"Enter the triangle width : ";

cin>>t[i].w;

}

int \*a= new int[n];

for(int i=0;i<n;i++)

{

a[i]=(t[i].h\*t[i].w)/2;

}

printArray(a, n);

mergeSort(a, 0, n - 1);

cout<<"The order of the areas of the triangles in increasing order of their areas :"<<endl;

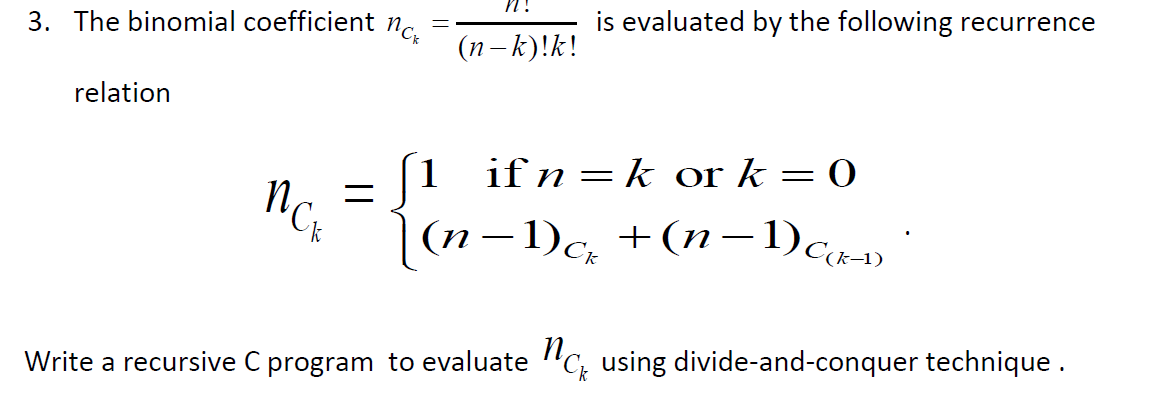
printArray(a, n);

}

*Output:*



*Question 3:*



*Code:*

#include<stdio.h>

#include<conio.h>

int save[100][100];

int coef(int n, int k)

{

if(n==k or k==0)

return 1;

else

return coef(n-1,k)+coef(n-1,k-1);

}

main(){

int n,k,m,i,j;

printf("Enter the values for n and k");

scanf("%d %d",&n,&k);

for(i=0;i<n;i++)

{for(j=0;j<n;j++)

save[i][j]=0;

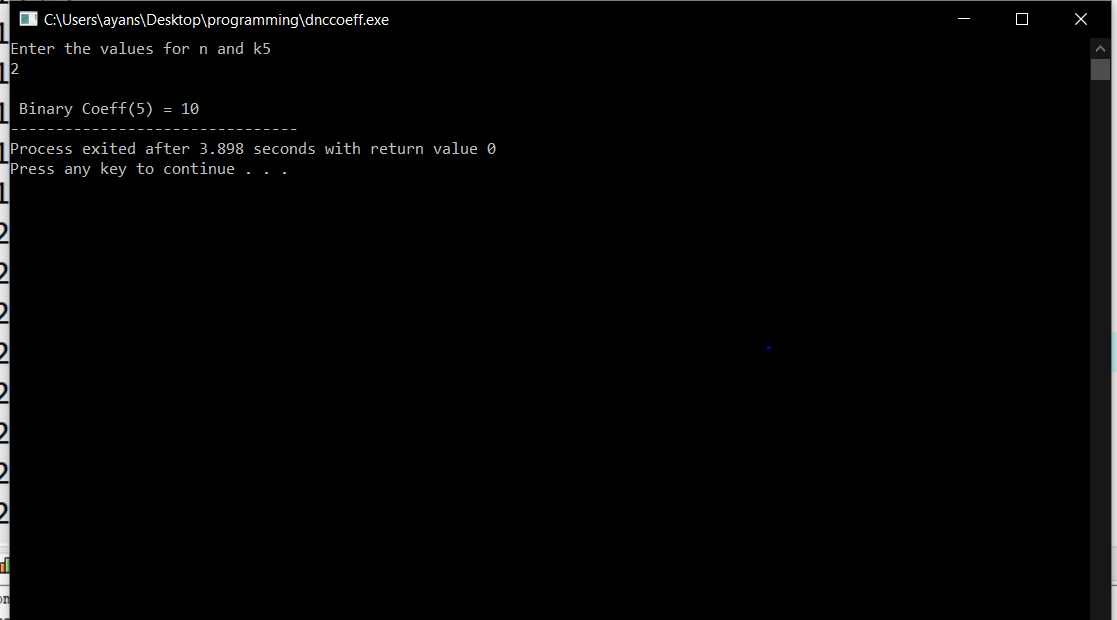
}

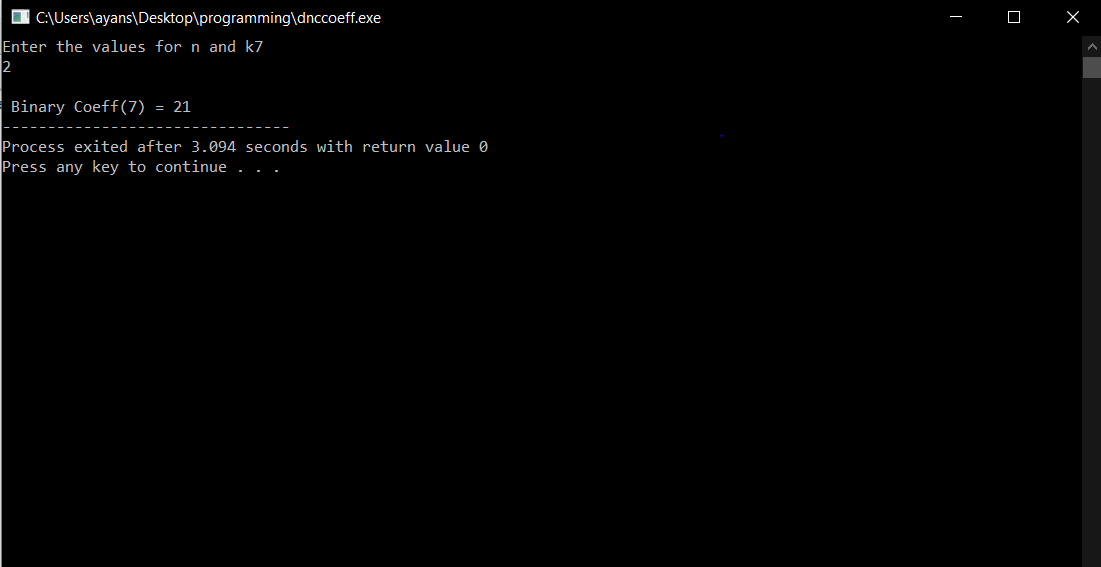
m=coef(n,k);

printf("\n Binary Coeff(%d) = %d",n,m);

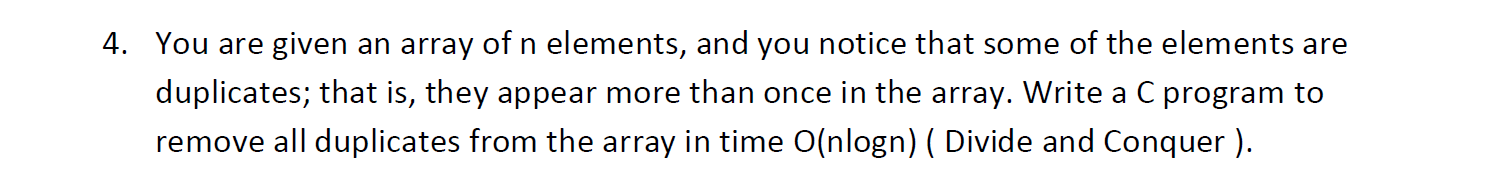
}

*Output:*





*Question 4:*



*Code:*

#include<iostream>

using namespace std;

int removeDuplicates(int arr[], int n)

{

if (n==0 || n==1)

return n;

int j = 0;

for (int i=0; i < n-1; i++)

if (arr[i] != arr[i+1])

arr[j++] = arr[i];

arr[j++] = arr[n-1];

return j;

}

void merge(int arr[], int p, int q, int r)

{

int i, j, k;

int n1 = q - p + 1;

int n2 = r - q;

int L[n1], R[n2];

for (i = 0; i < n1; i++)

L[i] = arr[p + i];

for (j = 0; j < n2; j++)

R[j] = arr[q + 1+ j];

i = 0;

j = 0;

k = p;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int p, int r)

{

if (p < r)

{

int q = (r+p)/2;

mergeSort(arr, p, q);

mergeSort(arr, q+1, r);

merge(arr, p, q, r);

}

}

void printArray(int A[], int size)

{

int i;

for (i=0; i < size; i++)

printf("%d ", A[i]);

printf("\n");

}

int main()

{

int n;

cout<<"Enter the length of the array :";

cin>>n;

int \*a=new int[n]();

for(int i=0;i<n;i++)

{

cout<<"Enter the element :";

cin>>a[i];

}

printArray(a, n);

mergeSort(a, 0, n - 1);

cout<<"Array after sorting with repeating elements :"<<endl;

printArray(a, n);

n = removeDuplicates(a, n);

cout<<"The array after removing the duplicates is :"<<endl;

for (int i=0; i<n; i++)

cout << a[i] << " ";

return 0;

}

*Output:*

