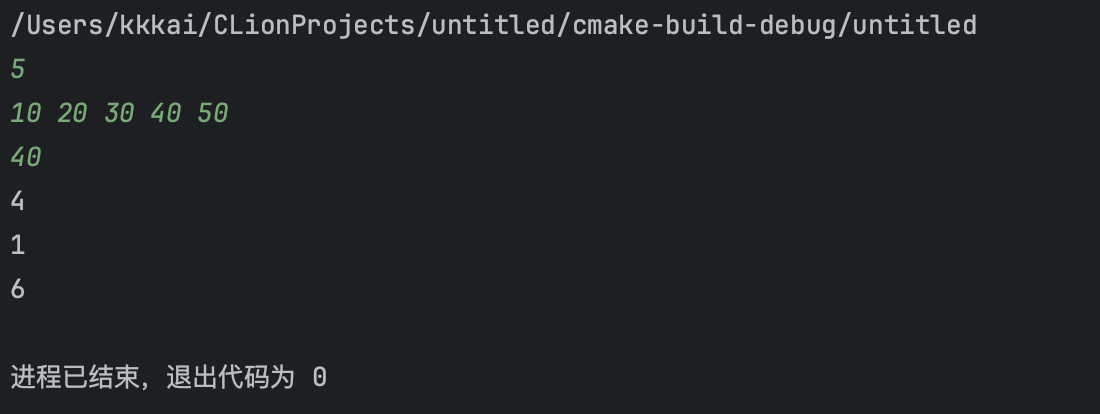
E2-1. Implement binary search and give some examples to test it. Input: a sorted array A of n distinct integers and an integer x Output: the index of x in array A

**//O(log(n))**

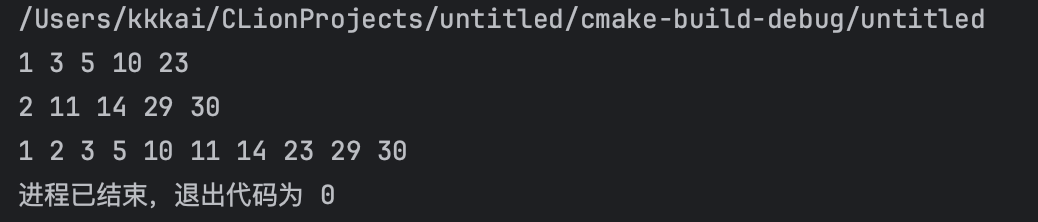
#include<iostream>  
#include<cmath>  
using namespace std;  
int binary\_search(int a[], int n, int x){  
 int u=floor(n/2);  
 while(a[u]!=x){  
 if(a[u]>x) u=floor(u/2);  
 else u=floor((u+n)/2);  
 }  
 return u;  
}  
int main()  
{  
 int n,x;  
 int a[1000001];  
 cin>>n;  
 for(int i=1;i<=n;i++)  
 cin>>a[i];  
 cin>>x;  
 cout<<binary\_search(a,n,x)<<endl;  
 int test1[]={1,2,3,4,5};  
 int test2[]={1,3,5,7,9,11,13,15};  
 cout<<binary\_search(test1,5,2)<<endl;  
 cout<<binary\_search(test2,8,13)<<endl;  
 return 0;  
}



E2-2. Merge two sorted lists and give some examples to test it. Input: two sorted lists A and B.  
Output: a sorted list which merges A and B.

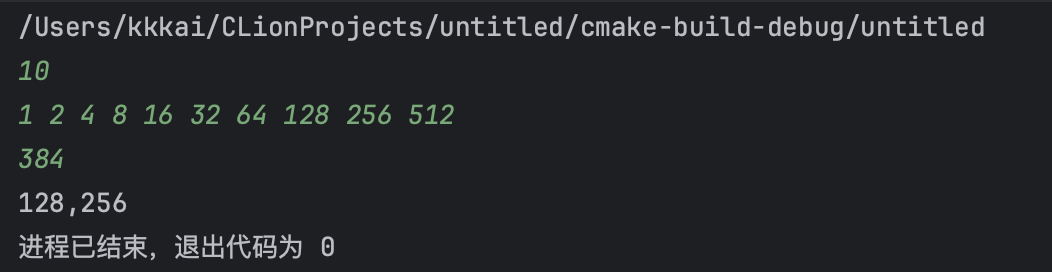
**//O(n)**

#include <iostream>  
using namespace std;  
void merge(int arr[], int tempArr[], int left, int mid, int right)  
{  
 int l\_pos = left;  
 int r\_pos = mid + 1;  
 int pos = left;  
  
 while (l\_pos <= mid && r\_pos <= right)  
 {  
 if (arr[l\_pos] < arr[r\_pos])  
 tempArr[pos++] = arr[l\_pos++];  
 else  
 tempArr[pos++] = arr[r\_pos++];  
 }  
 while (l\_pos <= mid)  
 tempArr[pos++] = arr[l\_pos++];  
 while (r\_pos <= right)  
 tempArr[pos++] = arr[r\_pos++];  
  
 while (left <= right) {  
 arr[left] = tempArr[left];  
 left++;  
 }  
  
}  
int main()  
{  
 int TEST1[] = {1,3,5,10,23};  
 int TEST2[] = {2,11,14,29,30};  
 int TEST[]={1,3,5,10,23,2,11,14,29,30};  
 for (int i : TEST1)  
 cout << i << " ";  
 cout << endl;  
 for (int i : TEST2)  
 cout << i << " ";  
 cout << endl;  
 int tempArr[10];  
 merge(TEST,tempArr,0,4,9);  
 for (int i : TEST)  
 cout << i << " ";  
  
 return 0;  
}

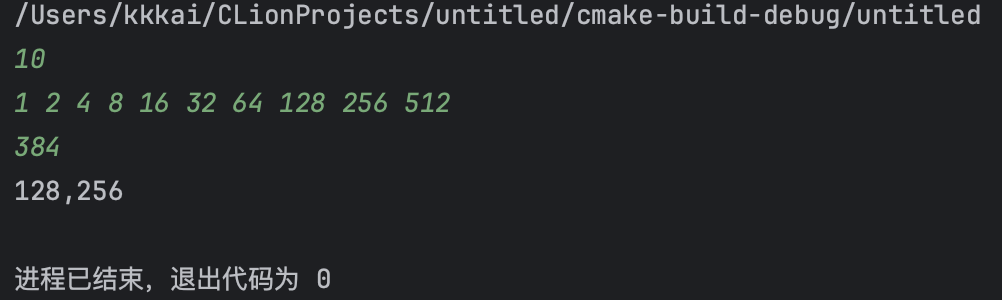


E2-3. Implement the algorithms of target-sum with O(n2), O(nlog(n)), O(n) respectively, and give some examples to test it.  
Input: a sorted array of n distinct integers an integer T.  
Output: two integers that sum to exactly T.

**//O(n^2)**  
#include<iostream>  
using namespace std;  
int main()  
{  
 int n,T;  
 cin>>n;  
 int a[100001];  
 for(int i=1;i<=n;i++)  
 cin>>a[i];  
 cin>>T;  
 for(int i=1;i<=n;i++)  
 for(int j=1;j<=n;j++)  
 if(a[i]!=a[j]&&a[i]+a[j]==T)  
 {cout<<a[i]<<","<<a[j];return 0;}  
 return 0;  
}

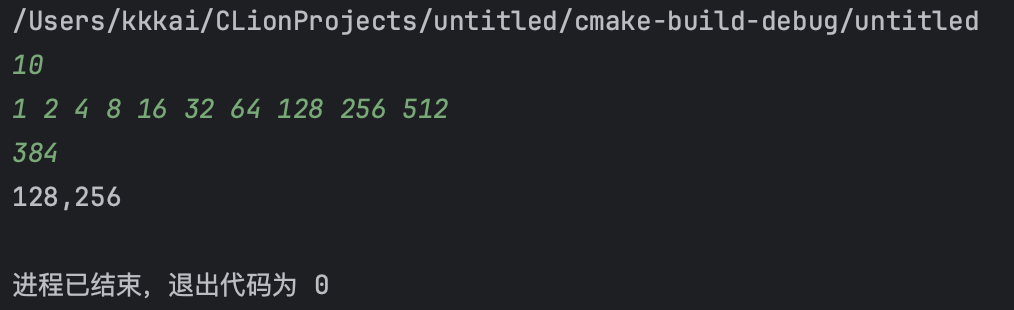


**//O(nlog(n))**  
#include <iostream>  
using namespace std;  
int binary\_search(int a[], int left, int right, int target) {  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
 if (a[mid] == target) {  
 return mid;  
 } else if (a[mid] < target) {  
 left = mid + 1;  
 } else {  
 right = mid - 1;  
 }  
 }  
 return -1;   
}  
int main() {  
 int n, T;  
 cin >> n;  
 int a[100001];  
 for (int i = 1; i <= n; i++)  
 cin >> a[i];  
 cin >> T;  
 for (int i = 1; i <= n; i++) {  
 int secondNum = T - a[i];  
 int index = binary\_search(a, 1, n, secondNum);  
 if (index != -1 && index != i) {  
 cout << a[i] << ',' << a[index] << endl;  
 break;  
 }  
 }  
 return 0;  
}



**//O(n)**

#include <iostream>  
using namespace std;  
int main()  
{  
 int n, T;  
 cin >> n;  
 int a[100001];  
 for (int i = 1; i <= n; i++)  
 cin >> a[i];  
 cin >> T;  
 int left = 1, right = n;  
 while (left < right)  
 {  
 int currentSum = a[left] + a[right];  
 if (currentSum == T)  
 {  
 cout << a[left] << ',' << a[right] << endl;  
 left++;  
 right--;  
 }  
 else if (currentSum < T) left++;  
 else right--;  
 }  
 return 0;  
}



E2-4. Implement the algorithms of the shortest distance and give some examples to test it.  
Input: a list of n points in the two-dimensional space {(x1,y1),...,(xn,yn)} Output: the pair that is closest to each other.

**//O(nlog(n))**

#include<iostream>  
#include<cmath>  
#include<algorithm>  
using namespace std;  
struct point  
{  
 double x, y;  
} p[200010];  
int n, temp[200010];  
pair<int, int> minDistancePoints; // Added to store the points with minimum distance  
bool cmp(const point &A, const point &B)  
{  
 if (A.x == B.x)  
 return A.y < B.y;  
 else  
 return A.x < B.x;  
}  
bool cmps(const int &a, const int &b)  
{return p[a].y < p[b].y;}  
  
double distance(int i, int j)  
{return sqrt((p[i].x - p[j].x) \* (p[i].x - p[j].x) + (p[i].y - p[j].y) \* (p[i].y - p[j].y));}  
  
double merge(int left, int right)  
{  
 double dis = 2 << 20;  
 if (left == right)  
 return dis;  
 if (left + 1 == right)  
 {  
 double d = distance(left, right);  
 if (d < dis)  
 {  
 dis = d;  
 minDistancePoints = {left, right};  
 }  
 return dis;  
 }  
 int mid = (left + right) >> 1;  
 double d1 = merge(left, mid);  
 double d2 = merge(mid + 1, right);  
 if (d1 < dis)dis = d1;  
 if (d2 < dis)dis = d2;  
 int k = 0;  
 for (int i = left; i <= right; i++)  
 {  
 if (fabs(p[i].x - p[mid].x) <= dis)  
 temp[k++] = i;  
 }  
  
 sort(temp, temp + k, cmps);  
  
 for (int i = 0; i < k; i++)  
 {  
 for (int j = i + 1; j < k && p[temp[j]].y - p[temp[i]].y < dis; j++)  
 {  
 double d = distance(temp[i], temp[j]);  
 if (d < dis)  
 {  
 dis = d;  
 minDistancePoints = {temp[i], temp[j]};  
 }  
 }  
 }  
 return dis;  
}  
int main()  
{  
 cin >> n;  
 for (int i = 0; i < n; i++)  
 cin >> p[i].x >> p[i].y;  
 sort(p, p + n, cmp);  
 double minDist = merge(0, n - 1);  
 cout << "(" << p[minDistancePoints.first].x << "," << p[minDistancePoints.first].y << "),("  
 << p[minDistancePoints.second].x << "," << p[minDistancePoints.second].y << ")" << endl;  
 return 0;  
}

