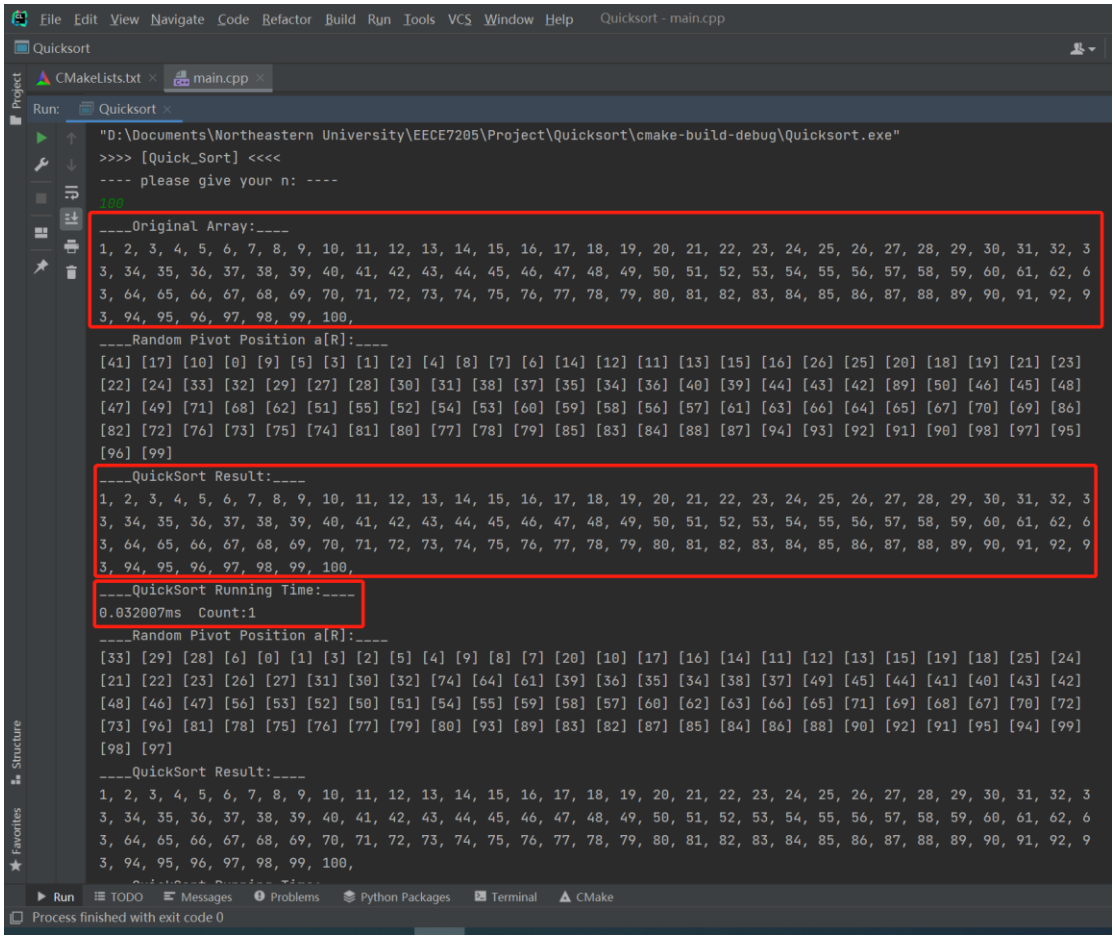
HW2_YIXIAO CHEN_002198256

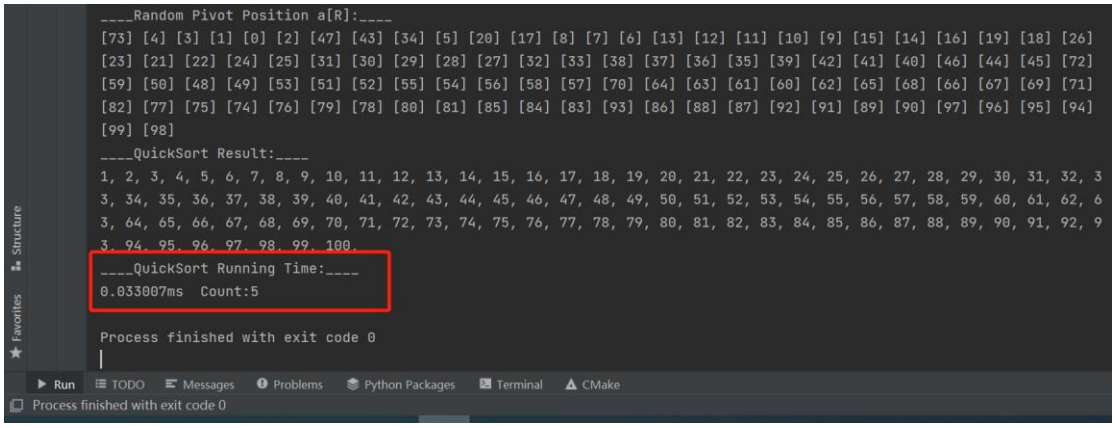
Q1:

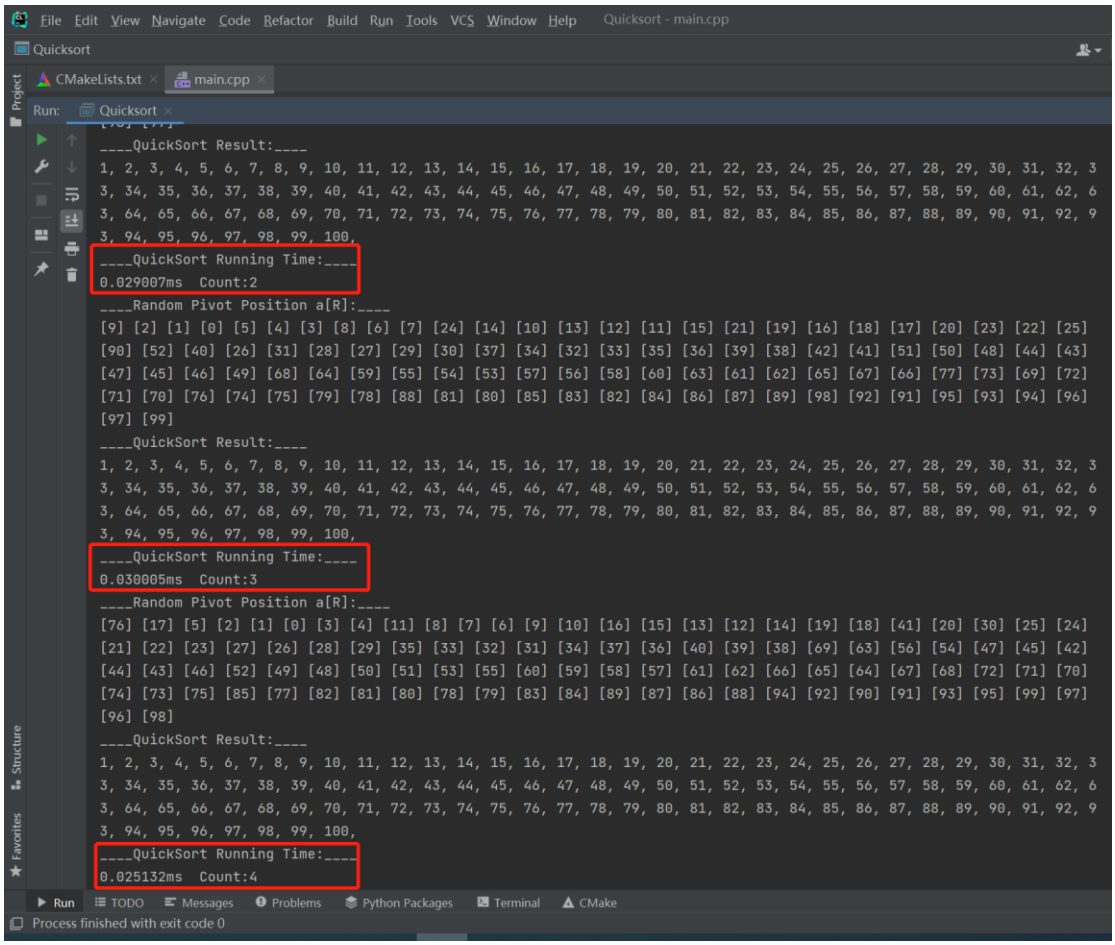
Question 1 (2 pt.) Randomized Quicksort: Write codes for randomized quicksort. You may need rand() to generate random numbers. Run the randomized quicksort 5 times for input array A = {1, 2, 3, …, 99, 100} and report the 5 running times.

------**Codes** after result for every question below------

【program running result】







【randomized quicksort code】_by cyx

```
#include <iostream>
#include <ratio>
#include <chrono>
#include <random>
using namespace std;
using namespace chrono;
int partition(int a[], int l, int r)
{
    int R = rand()%(r - l + 1) + l;//随机生成[l,r]之间整数
    cout<< "[" << R << "] ";
    int pivot = a[R];
    a[R] = a[r];
    a[r] = pivot;
    int e;
    int i = l - 1;
    for(int j = l; j < r; j++)
    {
```

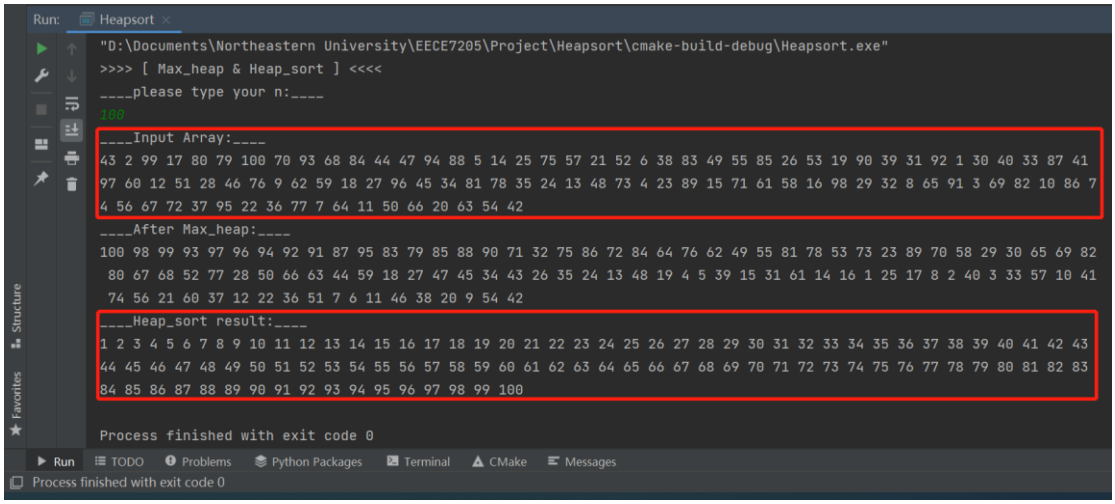
```
            if(a[j] <= pivot)
            {
                i = i + 1;
                e = a[i];
                a[i] = a[j];
                a[j] = e;
            }
        }
        a[r] = a[i + 1];
        a[i + 1] = pivot;
        return i+1;
}
void quicksort(int a[], int l, int r)
{
        int p;
        if(l <= r)
        {
            p = partition(a, l, r);
            quicksort(a, l, p-1);
            quicksort(a, p+1, r);
        }
}
void print_array(int a[], int n)
{
        for(int i = 0; i < n; i++)
        {
            cout << a[i] << ", ";
        }
        cout << endl;
}
int main() {
        cout << ">>>> [Quick_Sort] <<<<" << endl;
        int n;
        cout << "---- please give your n: ----" << endl;
        cin >> n;
        int * a, * v;
        a = new int[n];
        v = new int[n];
```

```
    for (int i = 0; i < n; i++)
    {
        a[i] = i + 1;
        v[i] = a[i];
    }
    cout << "____Original Array:____" << endl;
    print_array(v, n);
    for(int j = 1; j <= 5; j++)
    {
        cout << "____Random Pivot Position a[R]:____" << endl;
        high_resolution_clock::time_point t1 = high_resolution_clock::now();
        quicksort(a, 0, n-1);
        cout << endl;
        high_resolution_clock::time_point t2 = high_resolution_clock::now();
        duration<double, ratio<1, 1000>> duration_QS =
duration_cast<duration<double, ratio<1, 1000>>>(t2 - t1);
        cout << "____QuickSort Result:____" << endl;
        print_array(a, n);
        cout << "____QuickSort Running Time:____"<< endl;
        cout << duration_QS.count() << "ms" << "    Count:" << j << endl;
    }
    return 0;
}
```

Q2:

Question 2 (2pt.) Heapsort: Write codes for heapsort. The input array is a random permutation of A={1, 2, 3, …, 99, 100}. You should write codes to generate and print the random permutation first.

【program running result】



【heap sort code】_by cyx

```
#include <iostream>
//#include<cmath>
#include<random>
#include<algorithm>
using namespace std;
void max_heapify(int a[], int i, int n) //heap 按 max 规则调整
{
    int l = 2 * i;
    int r = 2 * i + 1;
    int largest;
    if(l <= n && a[l] > a[i]){
        largest = l ;
    }
    else{
        largest = i;
    }
    if(r <= n && a[r] > a[largest]){
        largest = r;
    }
    int value;
    if(largest != i){
```

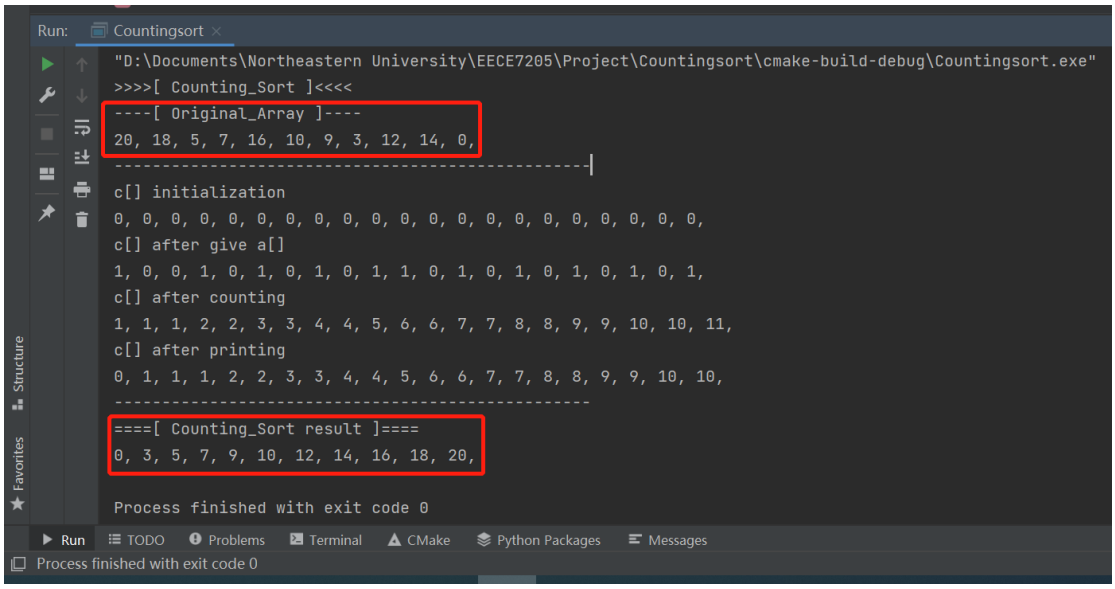
```
            value = a[i];
            a[i] = a[largest];
            a[largest] = value;
            max_heapify(a, largest, n);
        }
}
void build_max_heap(int a[], int n) //自身调用  构建 max hea{
        for (int i = n/2; i >= 1 ; i--)
        {
            max_heapify(a, i, n);
        }
}
void print(int a[], int n)    //打印输入序列{
        for (int i = 1; i <= n; i++)
        {
            cout << a[i] <<" ";
        }
        cout<< endl;
}
/*void print_heap_tree(int a[], int n){
        cout << "Max_Heap_Tree:" << endl;
        int total_level = log(n)/log(2);
        for(int l = 0; l <= total_level; l++)
        {
            for(int i = pow(2,l); i < pow(2,l+1) && i <= n ; i++)
            {
                if(a[i] <= 9)
                {
                    cout << a[i] <<"       ";
                }
                else
                {
                    cout << a[i] <<"      ";
                }
            }
            cout << endl;
        }
}*/
```

```
void heap_sort(int a[], int n){
	int * H;
	H = new int[n+1];
	for(int i = n; i >= 1; i--)
	{
		H[i] = a[1];
		a[1] = a[i];
		max_heapify(a, 1, i-1);
	}
	cout << "____Heap_sort result:____" << endl;
	print(H, n);
}
int main() {
	cout << ">>>> [ Max_heap & Heap_sort ] <<<<" << endl;
	int n;
	cout <<"____please type your n:____"<< endl;
	cin >> n; //读取目标数列总长度
	int * a;
	a = new int[n+1]; //数组指针从 0 开始，但 heap tree 里面 key 不能为 0
	a [0] = 0;//暂且设定数组 a[0]=0
	for (int i = 1; i <= n; i++)
	{
		a[i]= i;
	}
	for(int j = n; j >= 1; --j)
	{
		swap(a[j], a[rand()%j+1]);
	}
	cout << "____Input Array:____" << endl;
	print(a, n);
	build_max_heap(a, n);
	cout << "____After Max_heap:____" << endl;
	print(a, n);
	//print_heap_tree(a, n);
	heap_sort(a, n);
	return 0;
}
```

Q3:

Question 3 (1pt.) Counting Sort: Write codes for counting sort. The input array is A = {20, 18, 5, 7, 16, 10, 9, 3, 12, 14, 0}.

【program running result】



【Counting sort code】_by cyx

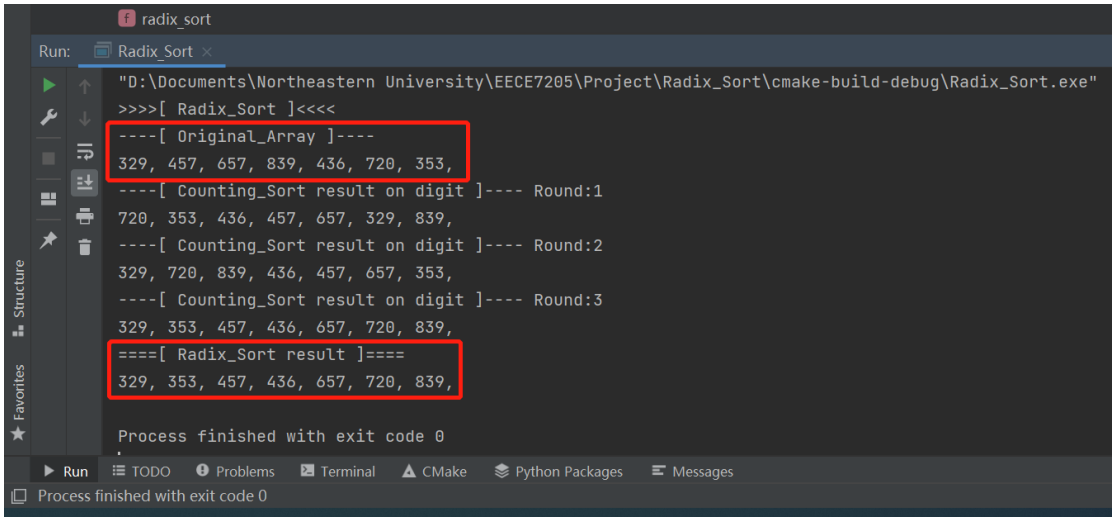
```
//#include<bits/stdc++.h>
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
void print(int a[], int n)    //打印输入序列{
        for (int i = 0; i < n; i++){
                cout << a[i] << ", ";
        }
        cout << endl;
}
void counting_sort(int a[], int n, int k){
        //int *b, *c;
        //b = new int[n];
        //c = new int[k + 1];
        int b[n];
        int c[k+1];
        memset(c,0, sizeof(c));
        //cout << "c[] initialization" << endl;
        //print(c, k+1);
```

```
    for(int j = 0; j < n; j++)
    {
        c[a[j]] = c[a[j]] + 1;
    }
    //cout << "c[] after give a[]" << endl;
    //print(c, k+1);
    for(int i = 1; i <= k; i++)//数组 C 计数从指针第二位（1）开始累加.
    {
        c[i] = c[i] + c[i-1];
    }
    //cout << "c[] after counting" << endl;
    //print(c, k+1);
    for(int j = n-1; j >= 0; j--)//此处需要考虑 A[0].
    {
        b[c[a[j]]-1] = a[j];
        c[a[j]] = c[a[j]] - 1;
    }
    //cout << "c[] after printing" << endl;
    //print(c, k+1);
    //cout << "--------------------------------------------------" <<endl;
    cout << "====[ Counting_Sort result ]====" << endl;
    print(b, n);
    //delete []b;
    //delete []c;
}
int main(){
    cout << ">>>>[ Counting_Sort ]<<<<" << endl;
    int a[11] = {20, 18, 5, 7, 16, 10, 9, 3, 12, 14, 0};
    int n = 11;
    int k = 20;
    cout << "----[ Original_Array ]----" << endl;
    print(a, n);
    //cout << "--------------------------------------------------" << endl;
    counting_sort(a, n, k);
    return 0;
}
```

Q4:

Question 4 (extra 1pt.) Radix Sort: Write codes for radix sort: use counting sort for decimal digits from the low order to high order. The input array is A = {329, 457, 657, 839, 436, 720, 353}.

【program running result】



【radix sort code】_by cyx

```
#include <iostream>
#include <cmath>
using namespace std;
void print(int a[], int n)    //打印输入序列
{
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << ", ";
    }
    cout << endl;
}
void radix_sort(int a[], int n, int k, int d)
{
    int *b, *c;
    b = new int[n];
    c = new int[k + 1];
    for(int dn = 10; dn <= pow(10,d); dn*=10)
    {
        for(int i = 0; i <= k; i++)//此处一定注意！课上伪代码数组取值范围 1-k; 实际情况数组各项取值范围 0-K.
```

```
            {
                c[i] = 0;
            }
            for(int j = 0; j < n; j++)//数组 C 各位的指针与数组 C 各位对应，所以当计算 A[j]个数的时候，对应 C 的位置应该+1.
            {
                c[(a[j]%dn/(dn/10))] = c[(a[j]%dn/(dn/10))] + 1;
            }
            for(int i = 1; i <= k; i++)//数组 C 计数从指针第二位（1）开始累加.
            {
                c[i] = c[i] + c[i-1];
            }
            for(int j = n - 1; j >= 0; j--)//此处需要考虑 A[0].
            {
                b[c[(a[j]%dn/(dn/10))]-1] = a[j];
                c[(a[j]%dn/(dn/10))] = c[(a[j]%dn/(dn/10))] - 1;
            }
            cout << "----[ Counting_Sort result on digit ]---- Round:"<< log10(dn) <<endl;
            print(b, n);
        }
        cout << "====[ Radix_Sort result ]====="<< endl;
        print(b, n);
}
int main()
{
        cout << ">>>>[ Radix_Sort ]<<<<" << endl;
        int a[] = {329, 457, 657, 839, 436, 720, 353};
        int n = 7;//number of elements in array a[]
        int k = 9;//max of each counting sort
        int d = 3;//relate to the number of digits
        cout << "----[ Original_Array ]----" << endl;
        print(a, n);
        radix_sort(a, n, k, d);
        return 0;
}
```