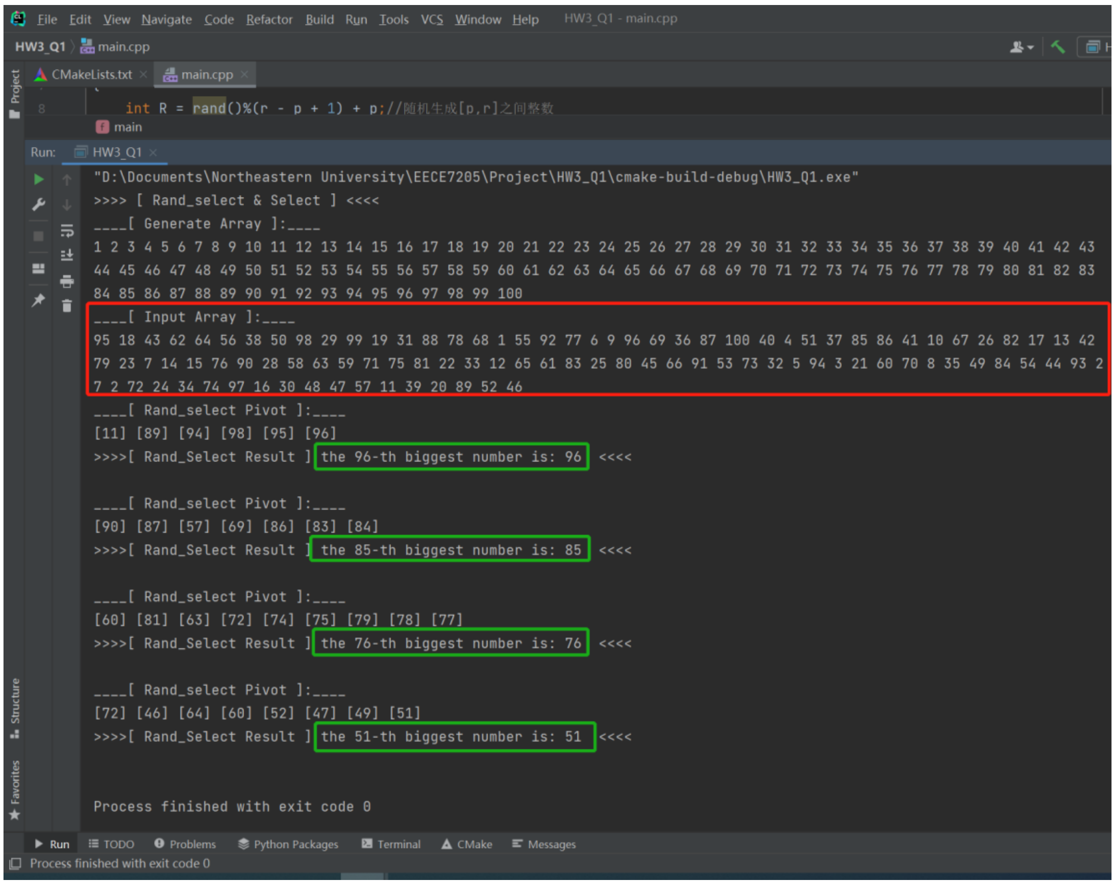
HW3_YIXIAO CHEN_002198256

Q1:

Question 1 (3 pt.) Order statistics: Write codes for Rand-Select (with linear expected running time) and Select (with linear worst-case running time). Test your two programs with an input array that is a random permutation of A = {1, 2, 3, …, 99, 100} (reuse of your Homework 2).

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

1.[Rand_select]:



```
    int R = rand()%(r - p + 1) + p;//随机生成[p,r]之间整数
```

```
"D:\Documents\Northeastern University\EECE7205\Project\HW3_Q1\cmake-build-debug\HW3_Q1.exe"
>>>> [ Rand_select & Select ] <<<<
____[ Generate Array ]:____
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
____[ Input Array ]:____
95 18 43 62 64 56 38 50 98 29 99 19 31 88 78 68 1 55 92 77 6 9 96 69 36 87 100 40 4 51 37 85 86 41 10 67 26 82 17 13 42
79 23 7 14 15 76 90 28 58 63 59 71 75 81 22 33 12 65 61 83 25 80 45 66 91 53 73 32 5 94 3 21 60 70 8 35 49 84 54 44 93 2
7 2 72 24 34 74 97 16 30 48 47 57 11 39 20 89 52 46
____[ Rand_select Pivot ]:____
[11] [89] [94] [98] [95] [96]
>>>>[ Rand_Select Result ] the 96-th biggest number is: 96  <<<<

____[ Rand_select Pivot ]:____
[90] [87] [57] [69] [86] [83] [84]
>>>>[ Rand_Select Result ] the 85-th biggest number is: 85  <<<<

____[ Rand_select Pivot ]:____
[60] [81] [63] [72] [74] [75] [79] [78] [77]
>>>>[ Rand_Select Result ] the 76-th biggest number is: 76  <<<<

____[ Rand_select Pivot ]:____
[72] [46] [64] [60] [52] [47] [49] [51]
>>>>[ Rand_Select Result ] the 51-th biggest number is: 51 <<<<


Process finished with exit code 0
```

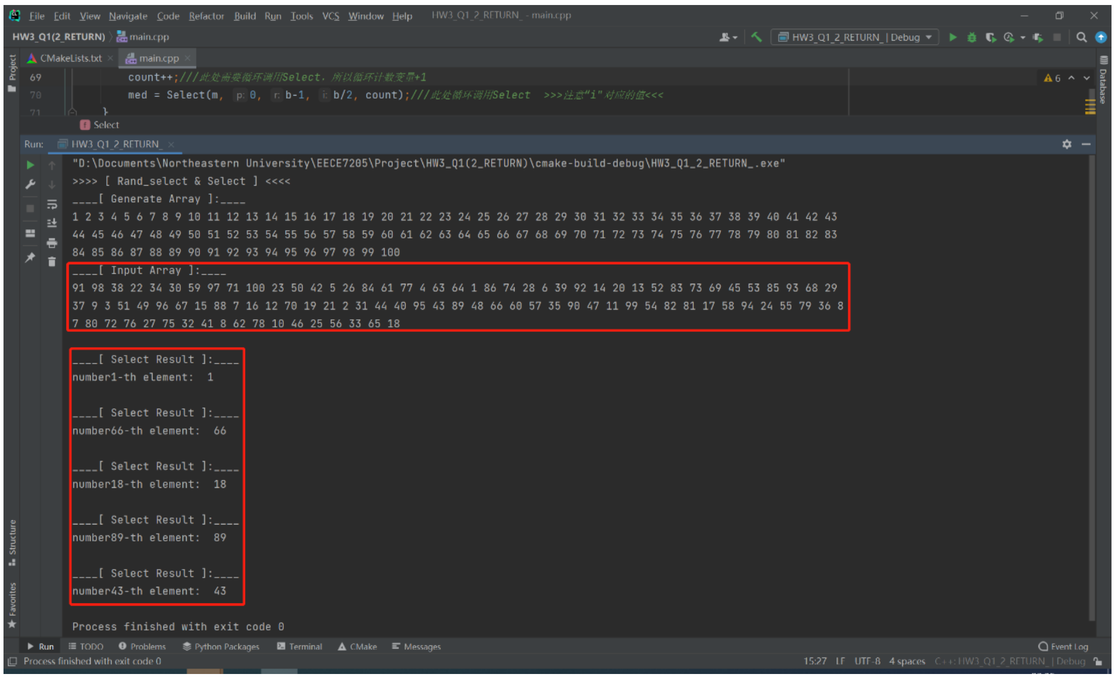
Process finished with exit code 0

【rand_select code】_by cyx

```
#include<iostream>
#include<random>
#include<algorithm>
#include<ctime>
using namespace std;
int Randomized_partition(int a[], int p, int r)
{
    int R = rand()%(r - p + 1) + p;//随机生成[p,r]之间整数
    cout<< "[" << a[R] << "] ";
    int pivot = a[R];
    swap(a[r], a[R]);
    int i = p - 1;
    for(int j = p; j < r; j++)
    {
        if(a[j] <= pivot)
        {
            i = i + 1;
            swap(a[i], a[j]);
        }
    }
    a[r] = a[i + 1];
    a[i + 1] = pivot;
    return i+1;
}
int Rand_Select(int a[], int p, int r, int i)
{
    int q, k;
    if(p ==r)
        return a[p];
    q = Randomized_partition(a, p, r);
    k = q - p + 1;
    if(i == k){
        return a[q];
    }else if(i < k){
        return Rand_Select(a, p, q-1, i);
    }else{
        return Rand_Select(a, q+1, r, i-k);//对于后半段序列 i 值已经发生变化
    }
}
void print(int a[], int n)    //打印输入序列
{
    for (int i = 0; i < n; i++)
    {
        cout << a[i] <<" ";
    }
    cout<< endl;
}
```

```
int main() {
        cout << ">>>> [ Rand_select & Select ] <<<<" << endl;
        srand((unsigned)time(NULL));
        int n = 100;
        int num;
        int result;
        //cout <<"____please type your n:____"<< endl;
        //cin >> num; //读取需要寻找的位数
        int * a;
        a = new int[n];
        for (int i = 0; i < n; i++)
        {
                a[i]= i+1;
        }
        cout << "____[ Generate Array ]:____" << endl;
        print(a, n);
        for(int j = n - 1 ; j >= 1; --j)
        {
                swap(a[j], a[rand()%j]);
        }
        cout << "____[ Input Array ]:____" << endl;
        print(a, n);
        for(int i=0; i<4; i++){
                num = rand()%100+1;
                cout << "____[ Rand_select Pivot ]:____" << endl;
                result = Rand_Select(a, 0, n-1, num);
                cout << "\n>>>>[ Rand_Select Result ] the " << num << "-th biggest number is: " <<
result << "    <<<<\n" << endl;
        }
        return 0;
}
```

2.[Select]:



```
count++;///此处需要循环调用Select，所以循环计数变量+1
med = Select(m, p: 0, r: b-1, i: b/2, count);///此处循环调用Select  >>>注意"i"对应的值<<<
}
```

```
"D:\Documents\Northeastern University\EECE7205\Project\HW3_Q1(2_RETURN)\cmake-build-debug\HW3_Q1_2_RETURN_.exe"
>>>> [ Rand_select & Select ] <<<<
____[ Generate Array ]:____
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
____[ Input Array ]:____
91 98 38 22 34 30 59 97 71 100 23 50 42 5 26 84 61 77 4 63 64 1 86 74 28 6 39 92 14 20 13 52 83 73 69 45 53 85 93 68 29
37 9 3 51 49 96 67 15 88 7 16 12 70 19 21 2 31 44 40 95 43 89 48 66 60 57 35 90 47 11 99 54 82 81 17 58 94 24 55 79 36 8
7 80 72 76 27 75 32 41 8 62 78 10 46 25 56 33 65 18

____[ Select Result ]:____
number1-th element:  1

____[ Select Result ]:____
number66-th element:  66

____[ Select Result ]:____
number18-th element:  18

____[ Select Result ]:____
number89-th element:  89

____[ Select Result ]:____
number43-th element:  43

Process finished with exit code 0
```

【Select code】_by cyx

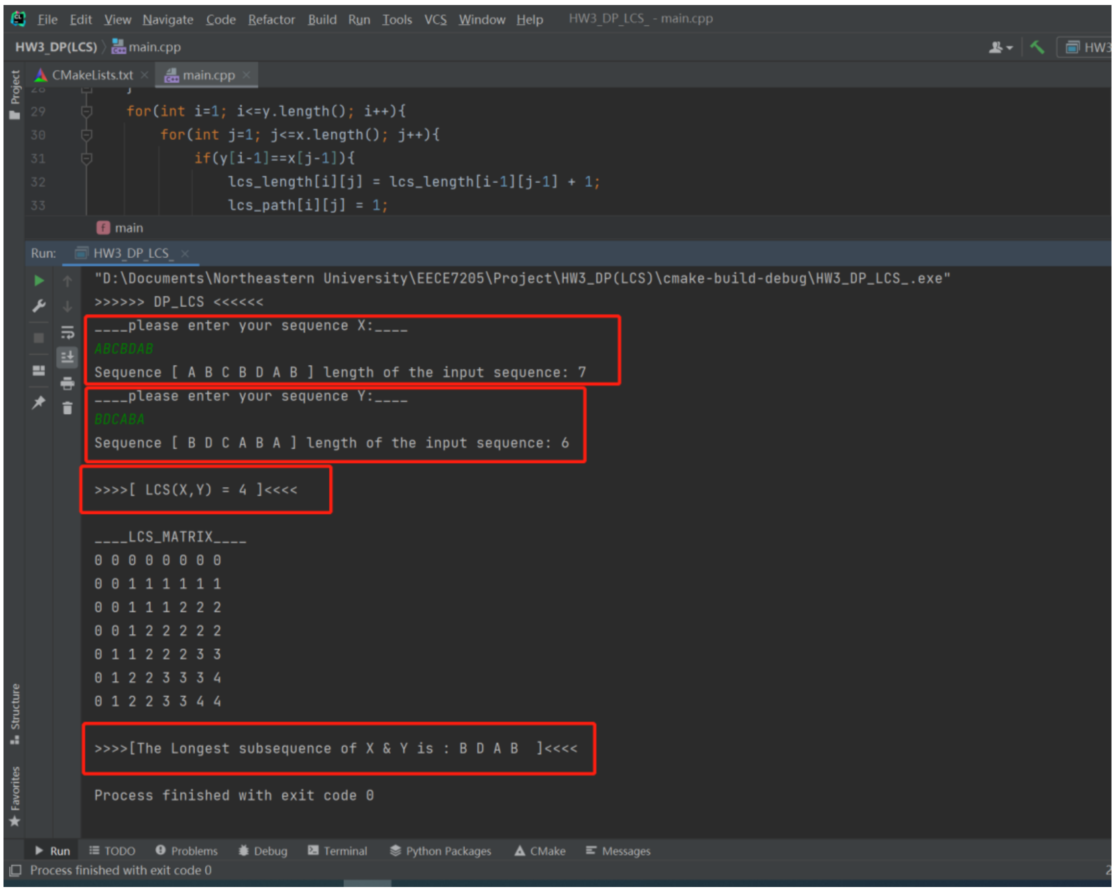
```
#include<iostream>
#include<random>
#include<algorithm>
#include<ctime>
using namespace std;
void Insertion_sort(int a[], int p, int r)///insertion 算法从之前代码复制过来时需要注意变量及使用问题
{
    if(p < r)
    {
        int value;
        int i, j;
        for (i = p + 1; i <= r; i++)///i 的终点是数列的终点，需要小于等于才能遍历整个数列
        {
            value = a[i];
            j = i - 1;
            while (j >= p && a[j] >= value)///此处需要特别注意 j 指针往回循环的终点是数列的起点
            {
                a[j + 1] = a[j];
                j--;
            }
            a[j + 1] = value;
        }
    }
}
int partition(int a[], int p, int r, int med)
{
    int pivot = med;///将之前选取的中位数座位 partition 函数的 pivot 值
    int x = a[r];///由于不知道中位数对应数组中的具体位置，所以暂时进行直接替换并且暂
存被替换的元素
    a[r] = med;
    int i = p - 1;///按照参考教材上 pivot 再最右侧的 partition 算法，i 和 j 分别从 p-1 和 p 开
始遍历数列
    for(int j = p; j < r; j++)
    {
        if(a[j] == pivot)///由于先前不知道中位数在原数组中的位置，所以此处单独与设立
条件（把被替换的 a[r]放回数组）
        {
            a[j] = x;
        }
        if(a[j] <= pivot)
        {
            i = i + 1;
            swap(a[i], a[j]);
        }
    }
    a[r] = a[i + 1];
    a[i + 1] = pivot;
    return i+1;
}
int Select(int a[], int p, int r, int i, int count)///需要循环调用 Select 以完成寻找中位数，count 变
量是 select 进行的次数
{
    int q, k, med;
    if(p == r)
        return a[p];
    if(r-p+1 <= 5)///考虑循环过程中数组长度小于等 5 的情况——base case
    {
        Insertion_sort(a,p,r);
        if(count > 1)///返回中位数 数组 的中位数
            return med = a[(r+p)/2];
        else///找到第 i 小的值了，直接 return 到主函数
            return a[p+i-1];
    }
    else{
        for (int j = p; j <= (r-5); j = j+5){
            Insertion_sort(a, j, j+4);///每 5 个数一组进行 insert 排序
        }
        int b = (r-p+1)/5;///新数组的长度就是原数组长度除以 5 取整
        int m[b];
        for (int j = 0; j < b; j++){
            m[j] = a[p + 2 + j*5];///按指针寻找将所有中位数写入新的数组，以进行下一轮
Select
        }
        count++;///此处需要循环调用 Select，所以循环计数变量+1
        med = Select(m, 0, b-1, b/2, count);///此处循环调用 Select   >>>注意“i"对应的值<<<
    }
    if(count > 1){
        count--;///需要回溯到最初状态 寻找最初状态的 a[]和对应的左右边界指针 p,r
        return med;///在回溯过程中一直返回先前找到的中位数，以进行接下来的 partition
工作
    }
    q = partition(a, p, r, med);
    k = q - p + 1;///此处以下部分逻辑与 rand——select 算法基本没有区别
    if(i == k){
        return a[q];
    }else if(i < k){
        return Select(a, p, q-1, i, 0);
    }else{
        return Select(a, q+1, r, i-k, 0);
    }
}
void print(int a[], int n)
{
    for (int i = 0; i < n; i++){
        cout << a[i] <<" ";
```

```
    }
    cout<< endl;
}
int main()
{
    srand((unsigned)time(NULL));///需要和 rand 一起使用，保证每次运行时候 rand 出来的值
有区别
    cout << ">>>> [ Rand_select & Select ] <<<<" << endl;
    int n = 100;
    int * a;
    a = new int[n];
    for (int i = 0; i < n; i++){
        a[i]= i+1;
    }
    cout << "____[ Generate Array ]:____" << endl;
    print(a, n);
    for(int j = n - 1 ; j >= 1; --j){
        swap(a[j], a[rand()%j]);///通过随机生成指针位置并交换 以达成打乱原有序数组的
目的
    }
    cout << "____[ Input Array ]:____" << endl;
    print(a, n);
    int num, S, count = 0;
    for(int i=0; i < 5; i++){
        num= rand()%100+1;
        S = Select(a, 0, n-1, num, count);
        cout << "\n____[ Select Result ]:____" << endl;
        cout <<"number"<< num <<"-th element:    " << S << endl;
    }
    return 0;
}
```

Q2:

Question 2 (2pt.) Dynamic Programming of LCS: Write codes for the longest common subsequence.

[DP_LCS result]:



```
    for(int i=1; i<=y.length(); i++){
        for(int j=1; j<=x.length(); j++){
            if(y[i-1]==x[j-1]){
                lcs_length[i][j] = lcs_length[i-1][j-1] + 1;
                lcs_path[i][j] = 1;
```

```
"D:\Documents\Northeastern University\EECE7205\Project\HW3_DP(LCS)\cmake-build-debug\HW3_DP_LCS_.exe"
>>>>>> DP_LCS <<<<<<
____please enter your sequence X:____
ABCBDAB
Sequence [ A B C B D A B ] length of the input sequence: 7
____please enter your sequence Y:____
BDCABA
Sequence [ B D C A B A ] length of the input sequence: 6

>>>>[ LCS(X,Y) = 4 ]<<<<

____LCS_MATRIX____
0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1
0 0 1 1 1 2 2 2
0 0 1 2 2 2 2 2
0 1 1 2 2 2 3 3
0 1 2 2 3 3 3 4
0 1 2 2 3 3 4 4

>>>>[The Longest subsequence of X & Y is : B D A B  ]<<<<

Process finished with exit code 0
```

Process finished with exit code 0

【Select code】_by cyx

```
#include <iostream>
using namespace std;
const int n = 100;
int lcs_length[n][n];
int lcs_path[n][n];
void print_lcs(string x, int i, int j)
{
    if(i==0 || j==0){
        return;
    }
    if(lcs_path[i][j]==1){
        print_lcs(x, i-1, j-1);
```

```
		cout << x[j-1] <<" ";
	}else if(lcs_path[i][j]==2){
		print_lcs(x, i-1, j);
	}else{
		print_lcs(x, i, j-1);
	}
}
int length_lcs(string x, string y)
{
	for(int i=1; i<=y.length(); i++){
		lcs_length[i][0] = 0;
	}
	for(int j=0; j<=x.length(); j++){
		lcs_length[0][j] = 0;
	}
	for(int i=1; i<=y.length(); i++){
		for(int j=1; j<=x.length(); j++){
			if(y[i-1]==x[j-1]){
				lcs_length[i][j] = lcs_length[i-1][j-1] + 1;
				lcs_path[i][j] = 1;
			}else if(lcs_length[i-1][j] >= lcs_length[i][j-1]){
				lcs_length[i][j] = lcs_length[i-1][j];
				lcs_path[i][j] = 2;
			}else{
				lcs_length[i][j] = lcs_length[i][j-1];
				lcs_path[i][j] = 3;
			}
		}
	}
	return lcs_length[y.length()][x.length()];
}
void print_matrix_lcs(string x, string y)
{
	for(int i=0; i<=y.length(); i++){
		for(int j=0; j<=x.length(); j++){
			cout << lcs_length[i][j] << " ";
		}
		cout << endl;
```

```
        }

}
void print(string a)
{
        cout << "Sequence [ ";
        for (int i=0; i<a.length(); i++){
                cout << a[i] << " " ;
        }
        cout << "] " << "length of the input sequence: " << a.length() << endl;
}
int main() {
        string a, b;
        int length;
        cout << ">>>>>> DP_LCS <<<<<<" << endl;
        cout << "____please enter your sequence X:____" << endl;
        cin >> a;
        print(a);
        cout << "____please enter your sequence Y:____" << endl;
        cin >> b;
        print(b);
        length = length_lcs(a, b);
        cout << "\n>>>>[ LCS(X,Y) = " << length << " ]<<<<\n" << endl;
        cout << "____LCS_MATRIX____" << endl;
        print_matrix_lcs(a,b);
        cout << "\n>>>>[The Longest subsequence of X & Y is : ";
        print_lcs(a, b.length(), a.length());
        cout << " ]<<<<" << endl;
        return 0;
}
```