UM-SJTU JOINT INSTITUTE
Data Structures and Algorithms
(VP281)

Programming Assignment

Programming Assignment Two
Linear-Time Selection

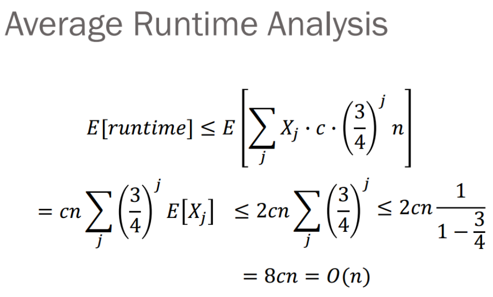
Name: Pan Chongdan
ID: 516370910121

Date: October 20, 2018

1 Introduction

The programming assignment asks me to implement two linear time selection algorithms including random selection and deterministic algorithm. The goal is clear, to gain experience in implementing these two sorting algorithms to find the n-th small number in a given random array.

Second, the assignment asks me to study the performance of these 2 studies by studying their time efficiency. In lecture slides 5, professor gives a summary for the time required to complete the selection.



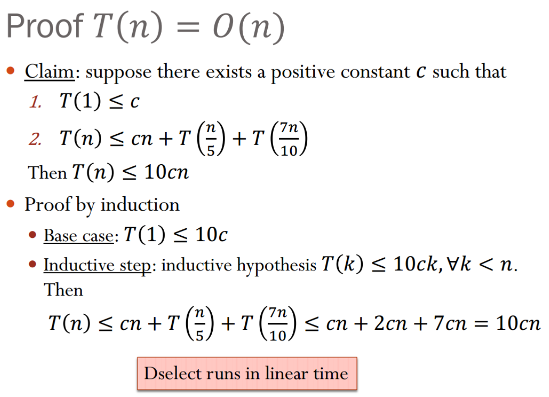
Average Runtime Analysis

$$E[runtime] \le E\left[\sum_j X_j \cdot c \cdot \left(\frac{3}{4}\right)^j n\right]$$

$$= cn\sum_j \left(\frac{3}{4}\right)^j E[X_j] \le 2cn\sum_j \left(\frac{3}{4}\right)^j \le 2cn\frac{1}{1-\frac{3}{4}}$$

$$= 8cn = O(n)$$

Figure 1: Random Selection



Proof $T(n) = O(n)$

- Claim: suppose there exists a positive constant c such that
  1. $T(1) \le c$
  2. $T(n) \le cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$

  Then $T(n) \le 10cn$
- Proof by induction
  - Base case: $T(1) \le 10c$
  - Inductive step: inductive hypothesis $T(k) \le 10ck, \forall k < n$. Then
    $$T(n) \le cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) \le cn + 2cn + 7cn = 10cn$$

Dselect runs in linear time

Figure 2: Deterministic Selection

However, we need to test the algorithms' time efficiency by ourselves, so I wrote a cpp file to print out the time required for each algorithm, which is included in appendix part.

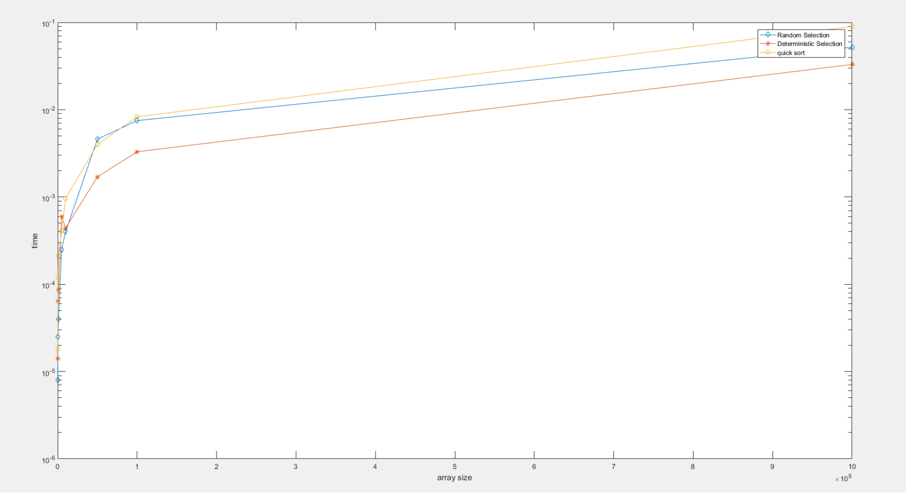
2 Result

To test time efficiency, I used clock() function to get the time required. To get rid of other disturbing factor, I used to variable start and stop to get the time right before and after the sorting complete, then their difference is the time used. In my analysis, I get 10 set of data ,from which the sorted array' size is 100, 500, 1000, 5000, 10000, 50000, 100000, 1000000.

Data Size	100	500	1000	5000	10000	50000	100000	1000000
Random Selection	8e-6	2.5e-5	4e-5	2.5e-4	4e-4	4.6e-3	7.5e-3	5.2e-2
Deterministic Selection	1.4e-5	6.4e-4	8.6e-4	6e-4	4.4e-4	1.7e-3	3.3e-3	3.3e-1
Quick Sort (in place)	1.8e-5	1.2e-4	2.1e-4	4.1e-4	9.6e-4	4.0e-3	8.3e-3	9e-2

3 Conclusion

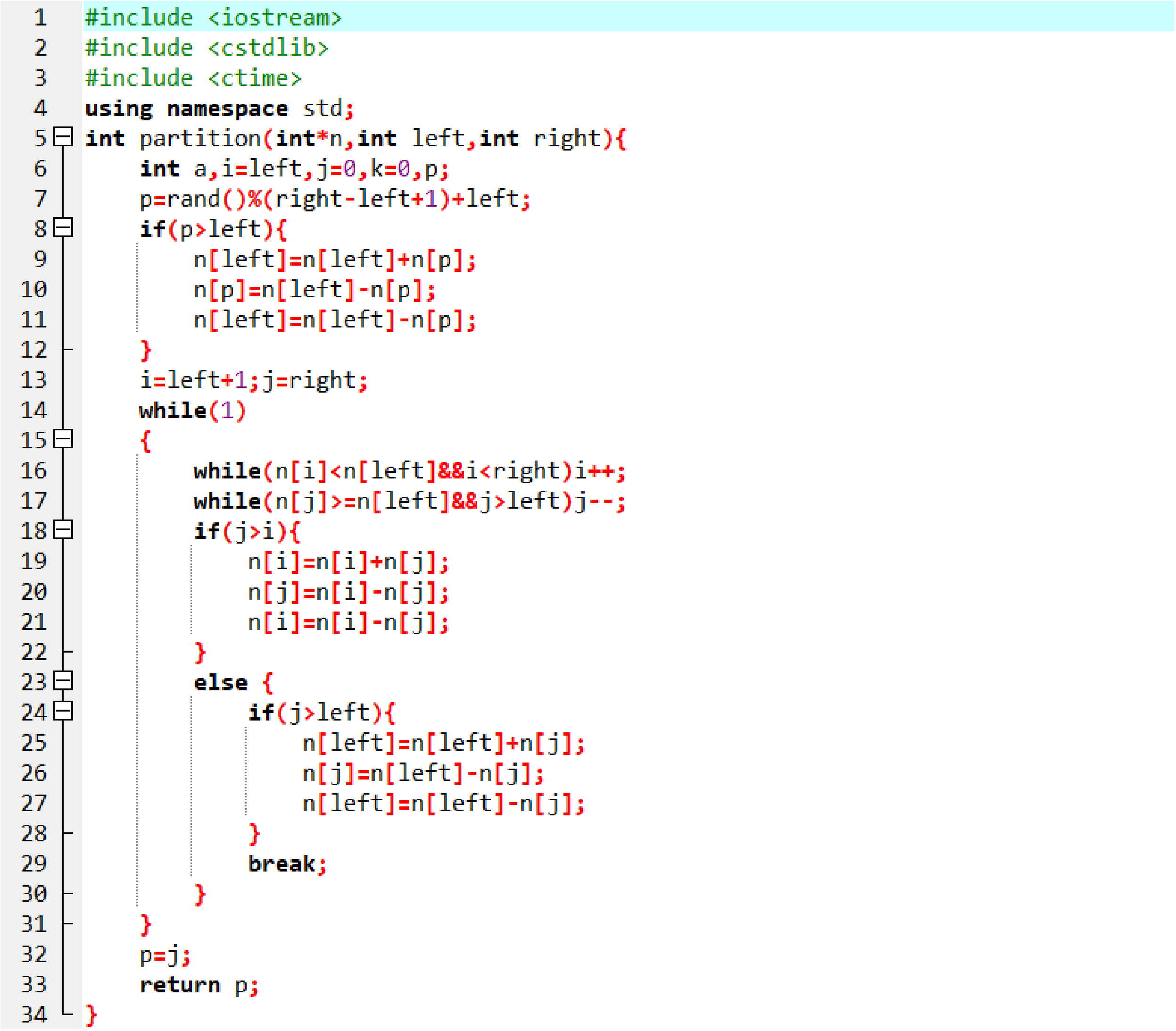
This chart is plot by matlab, showing the time efficiency of each algorithms compared to each other, and it has following characteristics.



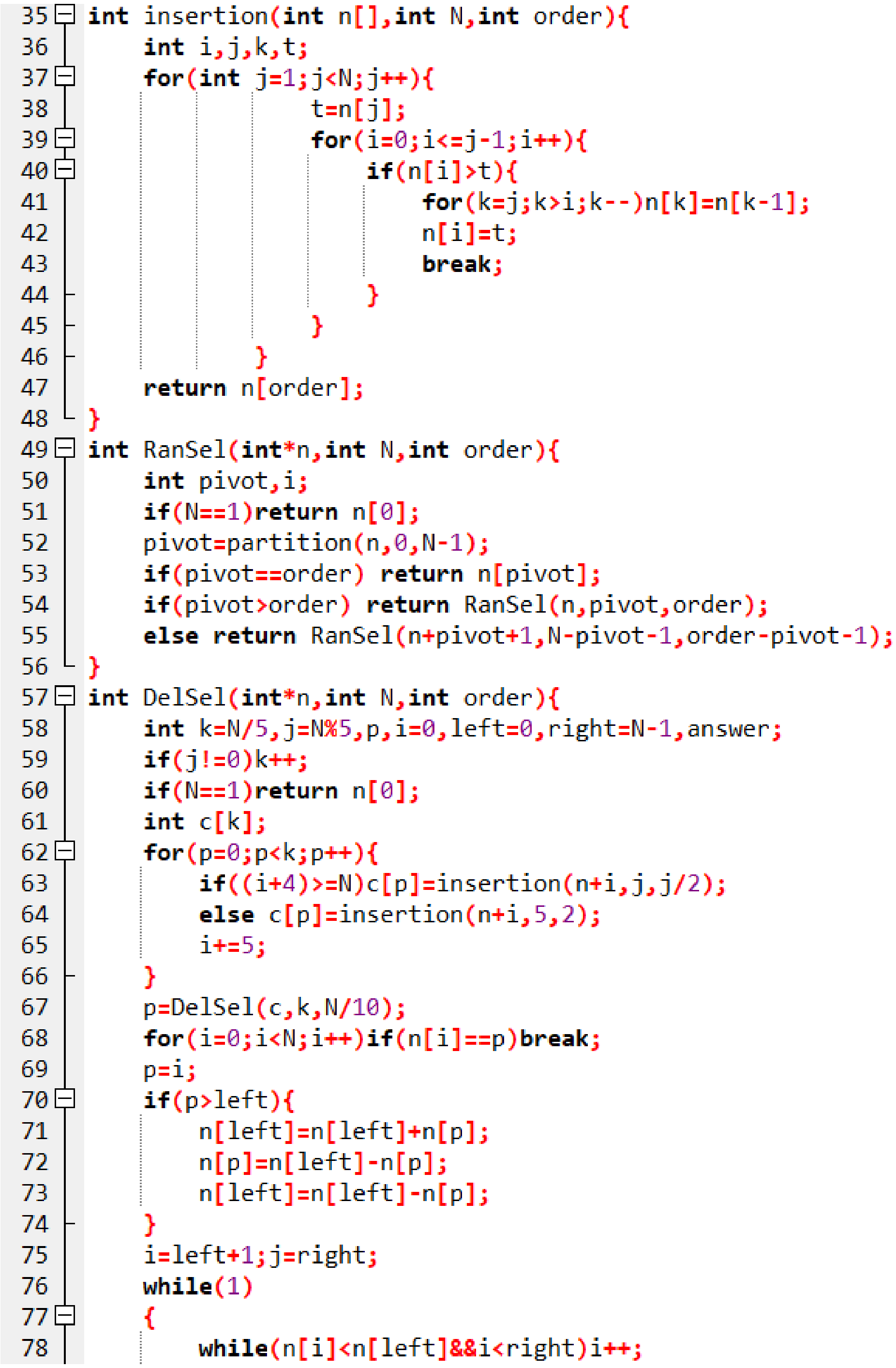
1. The average time required for each algorithm grows as the array grows bigger.
2. These three algorithm runs in a linear time when the array is big
3. As shown in the graph and lecture slides, deterministic selection requires the more time than the random selection and they're all quicker than quick sort when the array is big. Also these three ways are all Big O complexity.

4 Appendix

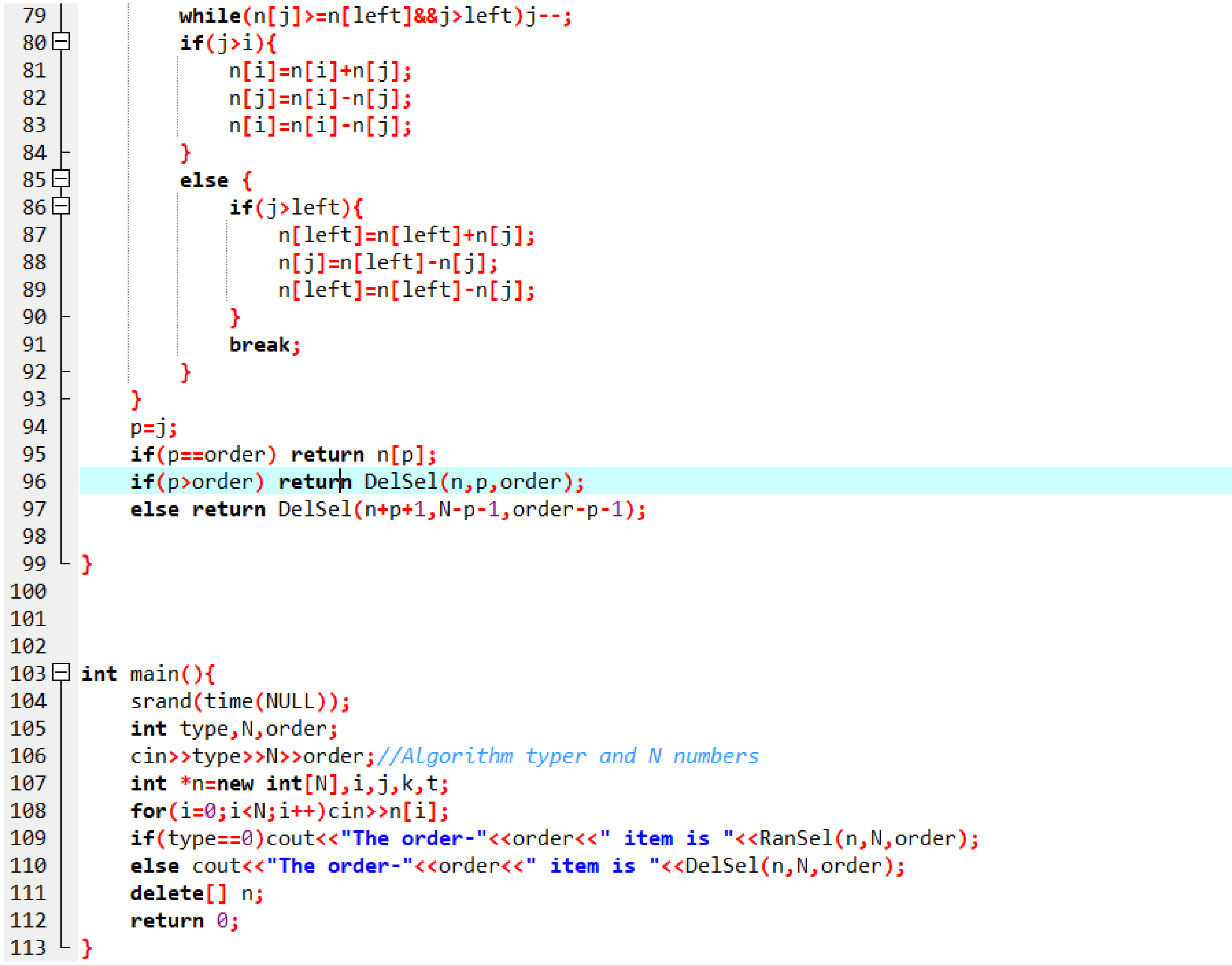
The appendix shows the cpp code for main project and time comparison.



```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int partition(int*n,int left,int right){
    int a,i=left,j=0,k=0,p;
    p=rand()%(right-left+1)+left;
    if(p>left){
        n[left]=n[left]+n[p];
        n[p]=n[left]-n[p];
        n[left]=n[left]-n[p];
    }
    i=left+1;j=right;
    while(1)
    {
        while(n[i]<n[left]&&i<right)i++;
        while(n[j]>=n[left]&&j>left)j--;
        if(j>i){
            n[i]=n[i]+n[j];
            n[j]=n[i]-n[j];
            n[i]=n[i]-n[j];
        }
        else {
            if(j>left){
                n[left]=n[left]+n[j];
                n[j]=n[left]-n[j];
                n[left]=n[left]-n[j];
            }
            break;
        }
    }
    p=j;
    return p;
}
```

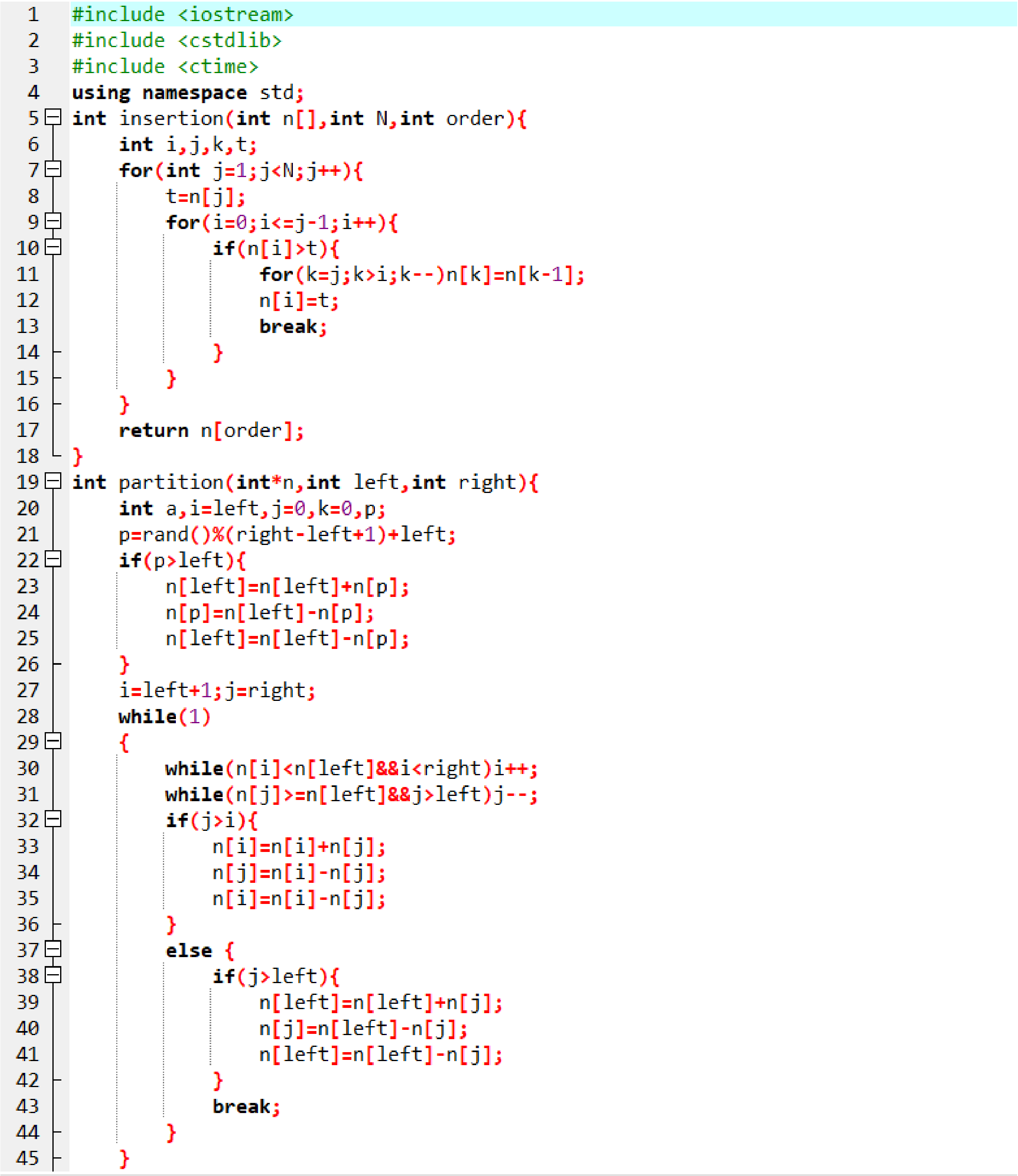


```
int insertion(int n[],int N,int order){
    int i,j,k,t;
    for(int j=1;j<N;j++){
                t=n[j];
                for(i=0;i<=j-1;i++){
                    if(n[i]>t){
                        for(k=j;k>i;k--)n[k]=n[k-1];
                        n[i]=t;
                        break;
                    }
                }
            }
    return n[order];
}
int RanSel(int*n,int N,int order){
    int pivot,i;
    if(N==1)return n[0];
    pivot=partition(n,0,N-1);
    if(pivot==order) return n[pivot];
    if(pivot>order) return RanSel(n,pivot,order);
    else return RanSel(n+pivot+1,N-pivot-1,order-pivot-1);
}
int DelSel(int*n,int N,int order){
    int k=N/5,j=N%5,p,i=0,left=0,right=N-1,answer;
    if(j!=0)k++;
    if(N==1)return n[0];
    int c[k];
    for(p=0;p<k;p++){
        if((i+4)>=N)c[p]=insertion(n+i,j,j/2);
        else c[p]=insertion(n+i,5,2);
        i+=5;
    }
    p=DelSel(c,k,N/10);
    for(i=0;i<N;i++)if(n[i]==p)break;
    p=i;
    if(p>left){
        n[left]=n[left]+n[p];
        n[p]=n[left]-n[p];
        n[left]=n[left]-n[p];
    }
    i=left+1;j=right;
    while(1)
    {
        while(n[i]<n[left]&&i<right)i++;
```

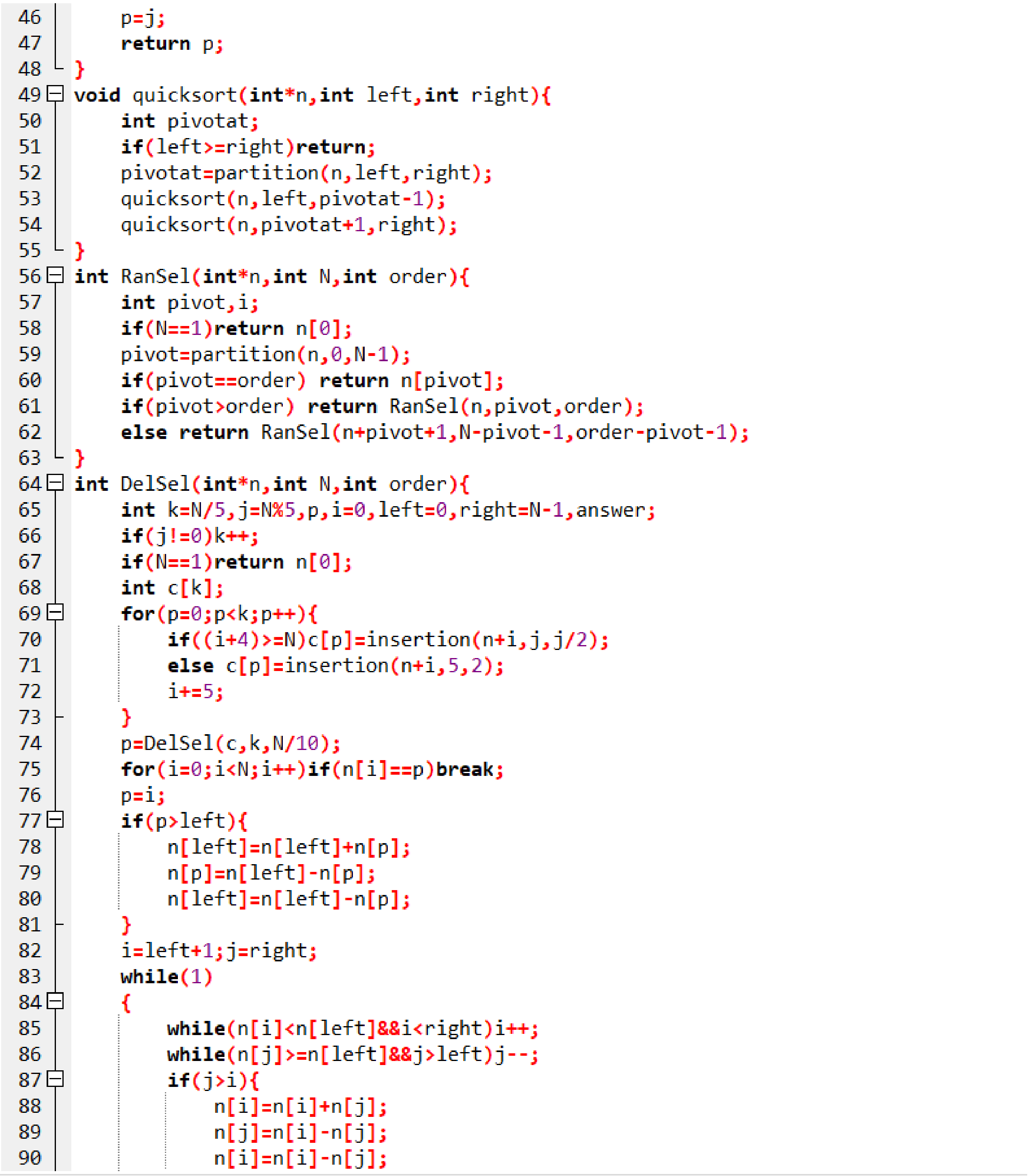


```
        while(n[j]>=n[left]&&j>left)j--;
        if(j>i){
            n[i]=n[i]+n[j];
            n[j]=n[i]-n[j];
            n[i]=n[i]-n[j];
        }
        else {
            if(j>left){
                n[left]=n[left]+n[j];
                n[j]=n[left]-n[j];
                n[left]=n[left]-n[j];
            }
            break;
        }
    }
    p=j;
    if(p==order) return n[p];
    if(p>order) return DelSel(n,p,order);
    else return DelSel(n+p+1,N-p-1,order-p-1);

}


int main(){
    srand(time(NULL));
    int type,N,order;
    cin>>type>>N>>order;//Algorithm typer and N numbers
    int *n=new int[N],i,j,k,t;
    for(i=0;i<N;i++)cin>>n[i];
    if(type==0)cout<<"The order-"<<order<<" item is "<<RanSel(n,N,order);
    else cout<<"The order-"<<order<<" item is "<<DelSel(n,N,order);
    delete[] n;
    return 0;
}
```

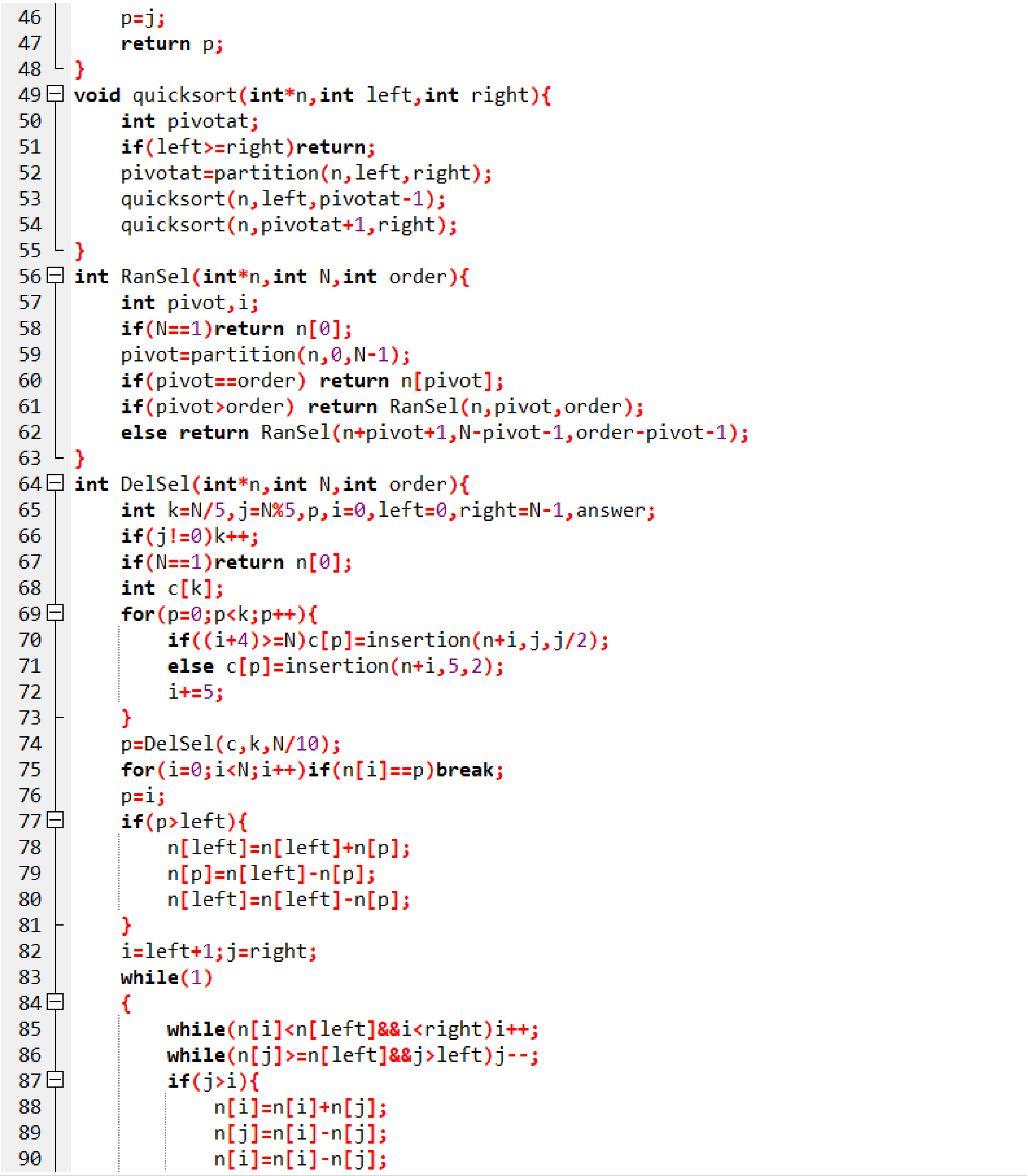
Then the appendix shows the cpp code for each sort algorithm



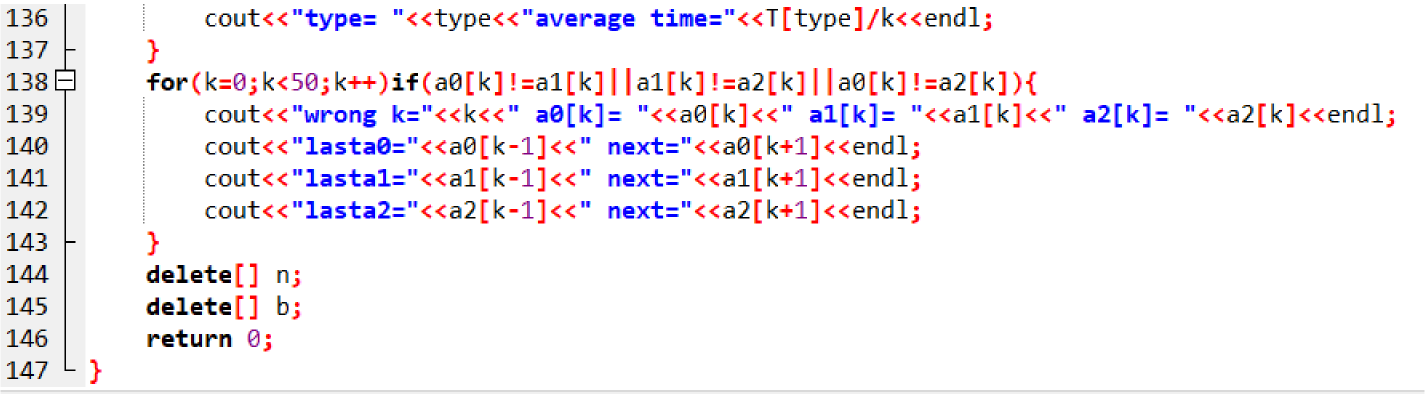
```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int insertion(int n[],int N,int order){
    int i,j,k,t;
    for(int j=1;j<N;j++){
        t=n[j];
        for(i=0;i<=j-1;i++){
            if(n[i]>t){
                for(k=j;k>i;k--)n[k]=n[k-1];
                n[i]=t;
                break;
            }
        }
    }
    return n[order];
}
int partition(int*n,int left,int right){
    int a,i=left,j=0,k=0,p;
    p=rand()%(right-left+1)+left;
    if(p>left){
        n[left]=n[left]+n[p];
        n[p]=n[left]-n[p];
        n[left]=n[left]-n[p];
    }
    i=left+1;j=right;
    while(1)
    {
        while(n[i]<n[left]&&i<right)i++;
        while(n[j]>=n[left]&&j>left)j--;
        if(j>i){
            n[i]=n[i]+n[j];
            n[j]=n[i]-n[j];
            n[i]=n[i]-n[j];
        }
        else {
            if(j>left){
                n[left]=n[left]+n[j];
                n[j]=n[left]-n[j];
                n[left]=n[left]-n[j];
            }
            break;
        }
    }
```



```
    p=j;
    return p;
}
void quicksort(int*n,int left,int right){
    int pivotat;
    if(left>=right)return;
    pivotat=partition(n,left,right);
    quicksort(n,left,pivotat-1);
    quicksort(n,pivotat+1,right);
}
int RanSel(int*n,int N,int order){
    int pivot,i;
    if(N==1)return n[0];
    pivot=partition(n,0,N-1);
    if(pivot==order) return n[pivot];
    if(pivot>order) return RanSel(n,pivot,order);
    else return RanSel(n+pivot+1,N-pivot-1,order-pivot-1);
}
int DelSel(int*n,int N,int order){
    int k=N/5,j=N%5,p,i=0,left=0,right=N-1,answer;
    if(j!=0)k++;
    if(N==1)return n[0];
    int c[k];
    for(p=0;p<k;p++){
        if((i+4)>=N)c[p]=insertion(n+i,j,j/2);
        else c[p]=insertion(n+i,5,2);
        i+=5;
    }
    p=DelSel(c,k,N/10);
    for(i=0;i<N;i++)if(n[i]==p)break;
    p=i;
    if(p>left){
        n[left]=n[left]+n[p];
        n[p]=n[left]-n[p];
        n[left]=n[left]-n[p];
    }
    i=left+1;j=right;
    while(1)
    {
        while(n[i]<n[left]&&i<right)i++;
        while(n[j]>=n[left]&&j>left)j--;
        if(j>i){
            n[i]=n[i]+n[j];
            n[j]=n[i]-n[j];
            n[i]=n[i]-n[j];
```



```
    p=j;
    return p;
}
void quicksort(int*n,int left,int right){
    int pivotat;
    if(left>=right)return;
    pivotat=partition(n,left,right);
    quicksort(n,left,pivotat-1);
    quicksort(n,pivotat+1,right);
}
int RanSel(int*n,int N,int order){
    int pivot,i;
    if(N==1)return n[0];
    pivot=partition(n,0,N-1);
    if(pivot==order) return n[pivot];
    if(pivot>order) return RanSel(n,pivot,order);
    else return RanSel(n+pivot+1,N-pivot-1,order-pivot-1);
}
int DelSel(int*n,int N,int order){
    int k=N/5,j=N%5,p,i=0,left=0,right=N-1,answer;
    if(j!=0)k++;
    if(N==1)return n[0];
    int c[k];
    for(p=0;p<k;p++){
        if((i+4)>=N)c[p]=insertion(n+i,j,j/2);
        else c[p]=insertion(n+i,5,2);
        i+=5;
    }
    p=DelSel(c,k,N/10);
    for(i=0;i<N;i++)if(n[i]==p)break;
    p=i;
    if(p>left){
        n[left]=n[left]+n[p];
        n[p]=n[left]-n[p];
        n[left]=n[left]-n[p];
    }
    i=left+1;j=right;
    while(1)
    {
        while(n[i]<n[left]&&i<right)i++;
        while(n[j]>=n[left]&&j>left)j--;
        if(j>i){
            n[i]=n[i]+n[j];
            n[j]=n[i]-n[j];
            n[i]=n[i]-n[j];
```



```
        cout<<"type= "<<type<<"average time="<<T[type]/k<<endl;
    }
    for(k=0;k<50;k++)if(a0[k]!=a1[k]||a1[k]!=a2[k]||a0[k]!=a2[k]){
        cout<<"wrong k="<<k<<" a0[k]= "<<a0[k]<<" a1[k]= "<<a1[k]<<" a2[k]= "<<a2[k]<<endl;
        cout<<"lasta0="<<a0[k-1]<<" next="<<a0[k+1]<<endl;
        cout<<"lasta1="<<a1[k-1]<<" next="<<a1[k+1]<<endl;
        cout<<"lasta2="<<a2[k-1]<<" next="<<a2[k+1]<<endl;
    }
    delete[] n;
    delete[] b;
    return 0;
}
```