Disk scheduling experiment

Experimental purpose and requirements:
Use high-level language to simulate the shortest seek time first algorithm and elevator scheduling algorithm in the disk scheduling algorithm. Requires input a sequence of disk access requests, outputs the order in which requests are actually processed, and calculates the output average seek

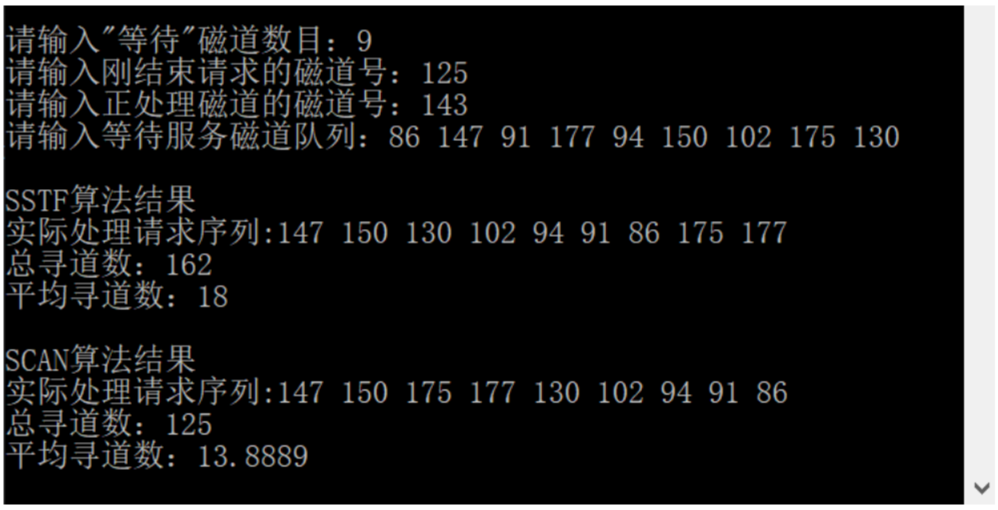
Experiment code:

```
#include <iostream>
using namespace std;

void swap1( int * left , int * right )
{
	int temp = * left ;
	*left = *right;
	*right = temp;
}
void SelectSort(int arr[], int num) {
	int i, j, Mindex;
	for (int i = 0; i < num; i++){
		Mindex = i;
		for (j = i + 1; j < num; j++) {
			if (arr[j] < arr[Mindex])
				Mindex = j;
		}
		swap1(& arr [i], & arr [Mindex]);
	}
}
//Shortest found time first (SSTF) algorithm
int SSTF( int tem_list [], int num ) {
	cout << "SSTF Algorithm Results" << endl << "Actually process the request sequence:" ;
	int *list;
	list = (int *)malloc(num * sizeof(int));
	for (int i = 0; i < num; i++) {
		list[i] = tem_list[i];
	}

	int now = list[0];
	int gap = abs(list[1] - list[0]);
```

```
    int all = 0;
    int flag = 1;

    for (int j = 0; j < num - 1; j++) { //
        for ( int i = 1; i < num ; i++) { //find the track number that can reach the minimum gap
            if (list[i] >= 0) {
                if (gap > abs(list[i] - now)) {
                    flag = i;
                    gap = abs(list[i] - now);
                }
            }
        }
        now = list[flag]; //The track number that has been confirmed in sequence makes it
invalid and makes it the new comparison base
        list[flag] = -1;
        all = all + gap;

        for ( int i = 1; i < num ; i++) { //reset gap
            if (list[i] >= 0) {
                gap = abs(list[i] - now);
                flag = i;
            }
        }
        cout << now << ' ';
    }
    cout << endl;

    float avg = float(all) / (num - 1);
    cout << "total seeks:  " << all << endl;
    cout << "Average seeks:" << avg << endl;
    //printf("Average number of seeks%.2f\n", all / (num - 1));
    return 0;
}
//Scan (SCAN) algorithm
int SCAN( int list [], int num , int pre ) {
    int *Outlist , k=1;
    Outlist = ( int *) malloc( num * sizeof ( int ));
    cout << "SCAN Algorithm Result" << endl << "Actually process the request sequence:" ;
    int first = list [0];
    Outlist[0] = first;
    int flag = -1;

    SelectSort( list , num ); // sorting algorithm
```

```
	for ( int i = 0; i < num ; i++) { // determine the position of the track number being processed
		if ( list [i] == first) {
			flag = i;
			break ;
		}
	}
	if (first > pre ) { //Such as from small to large
		for ( int i = flag + 1; i < num ; i++) {
			if ( list [i] > first) {
				//cout << list[i] << ' ';
				Outlist[k++] = list[i];
			}
		}
		for (int i = flag - 1; i >= 0; i--) {
			if (list[i] < first)
				//cout << list[i] << ' ';
				Outlist[k++] = list[i];
		}
	}
	else if (first < pre) { //如从大到小
		for (int i = flag - 1; i >= 0; i--) {
			if (list[i] < first)
				//cout << list[i] << ' ';
				Outlist[k++] = list [i];
		}
		for ( int i = flag + 1; i < num ; i++) {
			if ( list [i] > first) {
				//cout << list[i] << ' ';
				Outlist[k++] = list [i];
			}
		}
	}
	else { //If the current track number is the same as the previous track number, the input is
wrong
		cout << "ERROR, the previous track number is the same as the current track number"
<< endl;
		return 1;
	}

	int all=0;
	for ( int i = 1; i < num ; i++) { //Output the sequence of processing results and calculate the
total number of seeks
		cout << Outlist[i] << ' ' ;
		all = all + abs(Outlist[i] - Outlist[i - 1]);
```

```
        }
        cout << endl;

        float avg = float (all) / ( num - 1);
        cout << "Total Seeks:" << all << endl;
        cout << "Average seeks:" << avg << endl;
        //printf("Average number of seeks%.2f\n", all / (num - 1));
        return 0;
}

int main(){
        cout << "---------- Operating System Experiment 6 Experiment Date: 2019.6.12 -----------" <<
endl;

        int *list, num;
        cout << "Please enter \"Wait\"Number of tracks:" ;
        cin >> num; num++;
        list = ( int *)malloc(num * sizeof ( int ));
        cout << "Please enter the track number that just ended the request:" ;
        int pre; cin >> pre;
        cout << "Please enter the track number of the track being processed:" ;
        cin >> list[0];
        cout << "Please enter the track queue waiting to be serviced:" ;
        for ( int i = 1; i < num; i++) {
                cin >> list[i];
        }
        cout << endl;

        SSTF(list, num); //call SSTF algorithm
        cout << endl;
        SCAN(list, num, pre); //call SCAN algorithm

        return 0;
}
```



```
请输入"等待"磁道数目：9
请输入刚结束请求的磁道号：125
请输入正处理磁道的磁道号：143
请输入等待服务磁道队列：86 147 91 177 94 150 102 175 130

SSTF算法结果
实际处理请求序列:147 150 130 102 94 91 86 175 177
总寻道数：162
平均寻道数：18

SCAN算法结果
实际处理请求序列:147 150 175 177 130 102 94 91 86
总寻道数：125
平均寻道数：13.8889
```