```
#ifndef _ALL_H_
#define _ALL_H_


#define RAND_MAX 250
#include <vector>
#include <time.h>
#include <iostream>
#include <fstream>
using namespace std;


void selection(vector<char> & array);
int min(const vector<char> & array,int start);

void insertion(vector<char> & array);

void heap(vector<char> & array);
void add(vector<char> & array,char a);
char del(vector<char> & array);

void mergesort(vector<char> & array,int begin,int end);
void merge(vector<char> & array,int begin,int mid,int end);

void qsort(vector<char> & array,int begin,int end);
int partition(vector<char> & array,int begin,int end);

void rqsort(vector<char> & array,int begin,int end);

class Except{
};


#endif
```

```
#include "all.h"

void main(){

    ofstream output;
    output.open("select.xls",ios_base::app);

    for(int mycount = 1;mycount < 10;mycount++)
    {
        int number = 1000 * mycount;

        srand(1984);

        vector<char> * array = new vector<char>;

        for(int counter = 0;counter < number;counter++)
        array->push_back(rand());

        clock_t begintime = clock();
        if(begintime == clock_t(-1))
            throw Except();

        selection(*array);

        clock_t endtime = clock();
        if(endtime == clock_t(-1))
            throw Except();

        output << number << '\t' << double(endtime - begintime) << endl;

        delete array;
    }
    output.close();
}
```

```
#include "all.h"

void heap(vector<char> & array)
{
    vector<char> tmp;
    int end = array.size();
```



```
    for(int count = 0;count < end;count++)
        add(tmp,array[count]);

    int b0 = tmp[0],b1 = tmp[1],b2 = tmp[2],b3 = tmp[3],b4 = tmp[4];

    array.resize(0);

    for(count = 0;count < end;count++)
        array.push_back(del(tmp));

    int a0 = array[0],a1 = array[1],a2 = array[2],a3 = array[3],a4 = array[4];
}

void add(vector<char> & array,char a)
{
    array.push_back(a);
    int isbig = array.size() - 1;

    if (isbig == 0)
        return;

    while(isbig != 0 && array[isbig] < array[(isbig - 1) / 2])
    {
        swap(array[isbig],array[(isbig - 1) / 2]);
        isbig--;
        isbig/=2;
    }
}

char del(vector<char> & array)
{
    int end = array.size() - 1;
    int last = array[end];
    int retval = array[0];
    array.pop_back();

    if(end == 0)
        return retval;

    array[0] = last;
    int a = array[0];
    a = array.size();

    int where = 0;

    while((2 * where + 1 <= end - 1 && array[where] > array[2 * where + 1])
        || (2 * where + 2 <= end - 1 && array[where] > array[2 * where + 2]))
    {
        if(array[where] > array[2 * where + 1])
        {
            if(2 * where + 2 <= end - 1 && array[where] > array[2 * where + 2])
            {
                if(array[2 * where + 2] < array[2 * where + 1])
                {
                    swap(array[where],array[2 * where + 2]);
                    where *= 2;
                    where += 2;
                }
                else
                {
                    swap(array[where],array[2 * where + 1]);
                    where *= 2;
                    where++;
                }
            }
            else
            {
                swap(array[where],array[2 * where + 1]);
                where *= 2;
                where++;
            }
        }
        else if(2 * where + 2 <= end - 1)
        {
            swap(array[where],array[2 * where + 2]);
            where *= 2;
            where += 2;
```



```
        }
    }

    return retval;
}
```

```
#include "all.h"

void insertion(vector<char> & array){
    int end = array.size();
    int where;

    for(int count = 0;count < end;count++)
    {
        where = count;
        while(where > 0 && array[where - 1] > array[where])
        {
            swap(array[where - 1],array[where]);
            where--;
        }
    }
}
```

```
#include "all.h"

void mergesort(vector<char> & array,int begin,int end)
{
    if(begin >= end)
        return;

    int mid = (begin + end) / 2;
    mergesort(array,begin,mid);
    mergesort(array,mid + 1,end);
    merge(array,begin,mid,end);
}

void merge(vector<char> & array,int begin,int mid,int end)
{
    vector<char> tmp;
    int i = begin;
    int j = mid + 1;
    int tammat = end - begin + 1;
    int count;

    for(count = 0;count < tammat && i != mid + 1 && j != end + 1;count++)
    {
        int a = array[i];
        int b = array[j];
        if(array[i] < array[j])
        {
            tmp.push_back(array[i]);
            i++;
        }
        else
        {
            tmp.push_back(array[j]);
            j++;
        }
    }

    if(i != mid + 1)
    {
        int a = array[i];
        for(;count < tammat;count++)
            tmp.push_back(array[i++]);
    }

    if(j != end + 1)
    {
        int b = array[j];
        for(;count < tammat;count++)
            tmp.push_back(array[j++]);
    }

    int counter;

    for(count = begin,counter = 0;count <= end;count++,counter++)
```



```
        array[count] = tmp[counter];
}
```

```
#include "all.h"

void qsort(vector<char> & array,int begin,int end)
{
    if(end <= begin)
        return;

    int q = partition(array,begin,end);
    qsort(array,begin,q - 1);
    qsort(array,q + 1,end);
}

int partition(vector<char> & array,int begin,int end)
{
    char x = array[begin];
    int i = begin - 1;
    int j = end + 1;

    while(true)
    {
        do
        i++;
        while(array[i] <= x && i < j);

        do
        j--;
        while(array[j] > x && i < j);

        if(i < j)
            swap(array[i],array[j]);
        else
        {
            swap(array[begin],array[i - 1]);
            return i - 1;
        }
    }
}
```

```
#include "all.h"

void rqsort(vector<char> & array,int begin,int end)
{
    if(end <= begin)
        return;

    int w = rand() % (end - begin) + begin;
    swap(array[begin],array[w]);

    int q = partition(array,begin,end);
    rqsort(array,begin,q - 1);
    rqsort(array,q + 1,end);
}
```

```
#include "all.h"

void selection(vector<char> & array)
{
    int end = array.size();
    int where;

    for(int count = 0;count < end;count++)
    {
        where = min(array,count);
        if(where != count)
            swap(array[count],array[where]);
    }
}

int min(const vector<char> & array,int start)
{
    int end = array.size();
    int min = array[start];
    int retval = start;
```

```
    for(int count = start + 1;count < end;count++)
    {
        if(array[count] < min)
        {
            min = array[count];
            retval = count;
        }
    }

    return retval;
}
```
