＜演習課題＞　探索(二分探索)

【課題問題 1】

ソートされた数列を二分探索のアルゴリズムで探索したい．データは配列に入っていると考え，探索範囲のインデックスの最小値を pl 最大値を pr とする．また，pr と pl の中点を pc としてデータの二分を行う．下の初期データで，探索される過程を例に習って 2 回目以降を pl,pr,pc を明記して示しなさい．
まず、探索する値を 10 とした場合、探索が成功する過程を示しなさい。.
　探索終了時は「探索終了」と記述し、探索終了後の解答欄がある場合は「無し」と記入すること。

インデックス

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	2	3	5	6	7	9	10	11	12	15	16	17	18	20	21	23	25	26	28

例） 1 回目 :　pl=0,　pr=19,　pc=(pl+pr)/2=9.5->9,　a[9]=12 > key より pr=8 とする

解答）2 回目 : _____

解答）3 回目 : _____

解答）4 回目 : _____

解答）5 回目 : _____

解答）6 回目 : _____

次に、探索する値を 19 とした場合、探索が失敗する過程を示しなさい。
探索失敗時は「探索失敗」と記述し、探索失敗後の解答欄がある場合は「無し」と記入すること。

例） 1 回目 :　pl=0,　pr=19,　pc=(pl+pr)/2=9.5->9,　a[9]=12 < key より pl=10 とする

解答）2 回目 : _____

解答）3 回目 : _____

解答）4 回目 : _____

解答）5 回目 : _____

解答）6 回目 : _____

【課題問題 2】

キーボードから入力した 10 個のデータが昇順に並んでいなくても二分探索ができるようにしたい。ソースコード 1 のプログラムに、データを昇順に並べる関数を*（3）*の位置に、二分探索を行う関数を*（4）*の位置に追加することによって、入力データが昇順に並んでいなくても二分探索可能なプログラムを完成せよ。

なお、昇順に並び替える関数は単純挿入法(第 8 回演習参照)を使用し、関数名は insertion とすること。また、二分探索の関数名は bin_search とし、探索が成功した場合は探索された数の入っている配列の添字を返すものとし、探索が失敗する場合には-1 を返すものとする。メインプログラム中の*（1）には挿入法による整列処理*、*（2）*には昇順にソートした結果を表示する機能を追加すること。

ソースコード 1

```
#include <stdio.h>
#define NUM 10

void insertion(int a[], int n);
int bin_search(int a[], int n, int key);
int main(void)
{
        int     i,ky,idx;
        int     x[NUM];

        printf("Input integer number %d times ¥n",NUM);
        for (i=0;i<NUM;i++)     {
                printf("x[%d]:",i);
                scanf("%d",&x[i]);
        }

        （1）挿入法を用いて昇順にソートする。

        （2）挿入法のソート後に結果を表示する。

        printf("Number to search:");
        scanf("%d",&ky);

        idx=bin_search(x,NUM,ky);

        if (idx==-1)
                printf("Searching was failed!¥n");
        else
                printf("%d is located at %d ¥n",ky,idx);
        return(0);
}

void insertion(int a[], intn)
（3）挿入法の関数

int bin_search(int a[], int n, int key)
（4）二分探索の関数
```

【課題問題 3】

探索アルゴリズムの計算回数に注目して比較を行いたい。

2000 個のデータをファイルから読み込み、線形探索を行う場合と二分探索を行う場合の計算回数を比較する。二分探索を行う場合はデータをソートする必要があるので、探索前にクイックソートを使用してソートを行うものとする。なお、swap 関数はクイックソート関数で使用するものとする。作成するプログラムのメインプログラムおよび各種関数は下記ソースコード２の構成を参考にし、計算回数を算出する部分は各自プログラムに必要な機能を追加すること。

探索に使用するデータは/home0/staff/DataStruct/search のディレクトリにある test1.txt, test2.txt, test3.txt, test4.txt を各自ホームディレクトリにコピーして使用すること。各データファイルには 2000 個の整数データがランダムに近い順番に記録されているので、その中から 500(キー)を探索するプログラムを作成せよ。

比較した計算量をもとに、探索されたデータの位置による計算回数の違いについて各自考察を行うこと。なお、ここでいう計算回数とは、各探索法(線形探索および二分探索)における探索対象のデータとキーの比較回数を示すものとする。

ソースコード 2

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 2000
#define FMAX 20

void swap(int *x, int *y);
void quick(int a[],int left, int right);
int bin_search(int a[], int n, int key);
int lin_search(int a[], int n, int key);

int main(void)
{
        int     i;
        int     seisu[MAX];
        int     kosu,ky,idx0,idx1;
        char    fname[FMAX];
        FILE    *fp;

        kosu=MAX;                       /* データ個数を kosu に代入   */
        printf("Input file name:");
        scanf("%s",fname);              /* ファイル名の読み込み   */

        fp=fopen(fname,"r");            /* ファイルオープン   */

        if (fp==NULL)   {               /* ファイルオープン時のエラー処理   */
                printf("No file found !");
                exit(1);
        }

        （１）ファイルからデータを kosu 個読み込み配列 seisu に入れる。
```

```
        printf("Number to search:");
        scanf("%d",&ky);                    /*  キーを変数 ky に入力する　*/

        （２）線形探索を行い結果を変数 idx0 に出力する。

        （３）クイックソート及び二分探索を行い結果を変数 idx1 に出力する。

        （４）(２)の線形探索、(３)の二分探索にかかった計算回数を表示する。

        fclose(fp);                         /*  ファイルクローズ　*/
        return(0);
}

void swap(int *x, int *y)
（５）swap 関数

void quick(int a[],int left, int right)
（６）クイックソート関数

int bin_search(int a[], int n, int key)
（７）二分探索関数

int lin_search(int a[], int n, int key)
{
        int     i=0;

        while(1) {
                if (i==n)
                        return (-1);
                if (a[i]==key)
                        return (i);
                i++;
                }
}
```

以上