

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <urses.h>
#include <limits.h>

#define SIZE 10000 // Maximum number of data

typedef int key;

int n=100;// Number of data
key age[SIZE];

void show_menu();
void create_data();
void display_data();
// NO COMPARISONS SORTS
void address_calculation_sort();
// INTERCHANGE SORTS
void ordinary_bubble_sort();
void flagged_bubble_sort();
void shaker_sort();
void selection_sort();
// INTUITIVE SORTS
void insertion_sort();
void binary_insertion_sort();
void Shell_sort();// After D. Shell, insertion sort of data some
distance away
void radix_sort();
// RECURSIVE SORTS
void quick_sort();
// TREE SORTS
void heap_sort();
void binary_tree_sort();

main(){
    char ch;int i;
    initscr();
    cbreak();noecho();
    do{
        show_menu();ch=getch();for (i=0;i<SIZE;i++) age[i]=0;
        switch (ch) {
            case 'a' : clear();address_calculation_sort();break;
            case 'o' : clear();create_data();display_data(1);
                        ordinary_bubble_sort();display_data(2);break;
            case 'f' : clear();create_data();display_data(1);
                        flagged_bubble_sort();display_data(2);break;
            case 'k' : clear();create_data();display_data(1);
                        shaker_sort();display_data(2);break;
            case 's' : clear();create_data();display_data(1);
                        selection_sort();display_data(2);break;
            case 'i' : clear();create_data(1);display_data();
                        insertion_sort();display_data(2);break;

```

```

        case 'b' : clear();create_data();display_data(1);
binary_insertion_sort();display_data(2);break;
        case 'S' : clear();create_data();display_data(1);
                    Shell_sort();display_data(2);break;
        case 'r' : clear();create_data();display_data(1);
                    radix_sort();display_data(2);break;
        case 'q' : clear();create_data();display_data(1);
                    quick_sort();display_data(2);break;
        case 'h' : clear();create_data();display_data(1);
                    heap_sort();display_data(2);break;
        case 'x' : endwin();exit(1);
    }
    ch=getch();show_menu();
} while (1);
}

void show_menu(){
    clear();
    mvprintw(2,6,"0th location of the data array is NOT used");
    mvprintw(3,6,"NO COMPARISONS SORTS");
    attron(A_BOLD);mvprintw(4,10,"a");attroff(A_BOLD);
        printf("ddress_calculation_sort");
    mvprintw(5,6,"INTERCHANGE SORTS");
    attron(A_BOLD);mvprintw(6,10,"o");attroff(A_BOLD);
        printf("rdinary_bubble_sort\n");
    attron(A_BOLD);mvprintw(7,10,"f");attroff(A_BOLD);
        printf("lagged_bubble_sort\n");

    mvprintw(8,10,"sha");attron(A_BOLD);printf("k");attroff(A_BOLD);
        printf("er_sort\n");
    attron(A_BOLD);mvprintw(9,10,"s");attroff(A_BOLD);
        printf("election_sort\n");
    mvprintw(10,6,"INTUITIVE SORTS");
    attron(A_BOLD);mvprintw(11,10,"i");attroff(A_BOLD);
        printf("nsertion_sort\n");
    attron(A_BOLD);mvprintw(12,10,"b");attroff(A_BOLD);
        printf("inary_insertion_sort\n");
    attron(A_BOLD);mvprintw(13,10,"S");attroff(A_BOLD);
        printf("hell_sort\n");
    attron(A_BOLD);mvprintw(14,10,"r");attroff(A_BOLD);
        printf("adix_sort\n");
    mvprintw(15,6,"RECURSIVE SORTS");
    attron(A_BOLD);mvprintw(16,10,"q");attroff(A_BOLD);
        printf("uick_sort\n");
    mvprintw(17,6,"TREE SORTS");

    attron(A_BOLD);mvprintw(18,10,"h");attroff(A_BOLD);printf("eap_so
rt\n");

    mvprintw(19,10,"e");attron(A_BOLD);printf("x");attroff(A_BOLD);
        printf("it\n");
    refresh();
}

```

```

void create_data(){
    int i,j;
    int m=80; // Largest data
    // Random number between 1 and n is 1+(random()%n)
    for (i=1;i<=n;i++) {j=1+(random()%m);age[i]=(key)j;}
}

// NO COMPARISONS SORT

void address_calculation_sort(){
    int i,j;
    key data;
    char *message1="press any key to return to main menu";
    char *message2="Can sort data only between 1 and";
    nocbreak();echo();
    printf("%s %d\n",message2, SIZE-1);
    while (scanf("%d",&data),data>=1 && data <=SIZE-1){
        age[data]++;
    }
    for (i=1;i<=SIZE-1;i++)
        if (age[i]!=0) for (j=1;j<=age[i];j++) printf("%d ",i);

    printf("\n\n");
    printf("%s",message1);
    cbreak();noecho();refresh();
}

// INTERCHANGE SORTS

void ordinary_bubble_sort(){
    int j,k;
    key t;
    for (k=n-1;k>=1;k--)
        for (j=1;j<=k;j++)
            if (age[j]>age[j+1])
                {t=age[j];age[j]=age[j+1];age[j+1]=t;}
}

void flagged_bubble_sort(){
    int j,k,flag;
    key t;
    flag=n;
    while (flag!=0) {
        k=flag-1;flag=0;
        for (j=1;j<=k;j++)
            if (age[j]>age[j+1])
                {t=age[j];age[j]=age[j+1];age[j+1]=t;
                    flag=j;
                }
    }
}

void shaker_sort(){
    int j,l,r,flag;
    key t;

```

```

l=1;r=n-1;flag=n-1;
do {
    for (j=1;j<=r;j++)
        if (age[j]>age[j+1]) {
            t=age[j];age[j]=age[j+1];age[j+1]=t;flag=j;
        };
    r=flag-1;
    for (j=r;j>=1;j--)
        if (age[j]>age[j+1]) {
            t=age[j];age[j]=age[j+1];age[j+1]=t;flag=j;
        };
    l=flag+1;
} while (l<=r);
}

```

```

void Shell_sort(){
    int i,j,gap,jg;
    key t;
    gap=n/2;
    while (gap>0){
        for (i=gap+1;i<=n;i++) {
            j=i-gap;
            while (j>0) {
                jg=j+gap;
                if (age[j]<=age[jg]) j=0;
                else {t=age[j];age[j]=age[jg];age[jg]=t;};
                j=j-gap;
            }
        }
        gap=gap/2;
    }
}

```

```

void selection_sort(){
    int i,j,k;
    key t;
    for (i=1;i<=n-1;i++) {
        k=i;t=age[k];
        for (j=i+1;j<=n;j++) if (age[j]<t) {k=j;t=age[j];}
        age[k]=age[i];age[i]=t;
    }
}

```

// INTUITIVE SORTS

```

void insertion_sort(){
    int j,k;
    key t;
    age[0]=INT_MIN;
    for (j=2;j<=n;j++) {
        t=age[j];k=j-1;
        while (t<age[k]) { age[k+1]=age[k];k--;}
        age[k+1]=t;
    }
}

```

```

void binary_insertion_sort(){
    int j,l,m,r;
    key t;
    for (j=2;j<=n;j++) {
        t=age[j];l=1;r=j-1;
        while (l<=r) {
            m=(l+r)/2;
            if (t<age[m]) r=m-1;
            else l=m+1;
        }
        for (m=j-1;m>=l;m--) age[m+1]=age[m];
        age[l]=t;
    }
}

void radix_sort(){}

// RECURSIVE SORTS

void quick_sort(){}

// TREE SORTS

void heap_sort(){
void create_heap(int i,int m){
    int j;
    enum boolean {False=0,True};
    enum boolean done;
    key t;
    done=False;
    t=age[i];j=2*i;
    while (j<=m && !done) {
        if (j<m)
            if (age[j]<age[j+1]) j++;
        if (t>age[j]) done=True;
        else {age[j/2]=age[j];j=2*j;}
    }
    age[j/2]=t;
}

    int j;
    key t;
    for (j=n/2;j>=1;j--) create_heap(j,n);
    for (j=n-1;j>=1;j--){
        t=age[j+1];age[j+1]=age[1];age[1]=t;
        create_heap(1,j);
    }
}

void display_data(int j){
    char *message1="press any key to return to main menu";
    int i;
    for (i=1;i<=n;i++) printf("%6d  ",age[i]);

    printf("\n\n");
}

```

```
    if (j==2) printf("%s",message1);  
    refresh();  
}
```