```
// David Sa
use std::cmp;

fn main() {
    let initial = [66, 70, 52, 93, 44, 67, 47, 10, 11, 13, 94, 99, 51,
12];
    let mut to_sort;
    println!("initial:          {:?}",initial);

    to_sort  = initial.clone();
    bubble_sort(&mut to_sort);
    println!("bubble-sorted:    {:?}",to_sort);

    to_sort  = initial.clone();
    sel_sort(&mut to_sort);
    println!("selection-sorted: {:?}",to_sort);

    to_sort  = initial.clone();
    insert_sort(&mut to_sort);
    println!("insertion-sorted: {:?}",to_sort);

    println!();
    println!("unordered search:");
    report_search(44,unordered_search(44,&initial[..]));
    report_search(43,unordered_search(43,&initial[..]));

    println!();
    println!("binary search:");
    report_search(44,binary_search(44,&to_sort[..]));
    report_search(43,binary_search(43,&to_sort[..]));

    println!();
    println!("the min and max of initial are {:?}",
             min_max(&initial[..]));
}

/*
// NOTE!! The following will not compile: It needs lifetime
annotations.
// We'll fix this later on.
fn swap(x :&mut u32, y : &mut u32) {
    let t = x;
    x = y;
    y = t;
}
*/

fn bubble_sort(a : &mut [u32]) {
    let len = a.len();
    for i in 0..len {
```

```
        for j in 0..(len-i-1) {
            if a[j]>a[j+1] {
                // swap the values of a[j] and a[j+1]
                let t = a[j];
                a[j] = a[j+1];
                a[j+1] = t;
            }
        }
    }
}

fn report_search(x : u32, r : Option<usize>) {
    print!("\t {} ",x);
    match r {
        None    => { println!("not found"); },
        Some(i) => { println!("found at index {}",i); },
    }
}

fn unordered_search(x : u32, a : &[u32]) -> Option<usize> {
    for i in 0..a.len() {
        if x==a[i] { return Some(i); }
    }
    None
}

fn sel_sort(a : &mut [u32]) {
    for i in 0..a.len() {
        let mut minimum = i;
        for j in i..a.len() {
            if a[j] < a[minimum] {
                minimum = j;
            }
        }
        a.swap(i, minimum);
    }
}

fn insert_sort(a : &mut [u32]) {
    for i in 1..a.len() {
        let mut j = i;
        let mut to_insert = a[i];
        while (j > 0) && (a[j - 1] > to_insert) {
            a[j] = a[j - 1];
            j -= 1;
        }
        a[j] = to_insert;
    }
}
```

```
fn binary_search(x : u32, a : &[u32]) -> Option<usize> {
    let mut low_index = 0;
    let mut high_index = a.len() - 1;
    while low_index <= high_index {
        let middle_index = (low_index + high_index) / 2;
        if x < a[middle_index] {
            high_index = middle_index - 1;
        } else if x > a[middle_index] {
            low_index = middle_index + 1;
        } else {
            return Some(middle_index);
        }
    }
    None
}

fn min_max(a : &[u32]) -> (u32,u32) {
    let len = a.len();
    assert!(len>0);
    let mut min = u32::max_value();
    let mut max = 0;
    for i in 0..a.len() {
        if a[i] > max {
            max = a[i];
        }
        if a[i] < min {
            min = a[i];
        }
    }
    return (min, max);
}

// NOTE:
// cmp::min(a,b) returns the minimum of a and b
// cmp::max(a,b) returns the maximum of a and b


/*
OUTPUT:

initial:          [66, 70, 52, 93, 44, 67, 47, 10, 11, 13, 94, 99, 51,
12]
bubble-sorted:    [10, 11, 12, 13, 44, 47, 51, 52, 66, 67, 70, 93, 94,
99]
selection-sorted: [10, 11, 12, 13, 44, 47, 51, 52, 66, 67, 70, 93, 94,
99]
insertion-sorted: [10, 11, 12, 13, 44, 47, 51, 52, 66, 67, 70, 93, 94,
99]

unordered search:
```

```
         44 found at index 4
         43 not found

binary search:
         44 found at index 4
         43 not found

the min and max of initial are (10, 99)

*/
```