

ENCM 339 Fall 2005:
Handout for T02 (Dr. Norman's tutorial)
Tues., Dec. 6

Author: Dr. S. A. Norman.

Electronic copies of handouts for L02 and T02 can be found at
<http://www.ene1.ucalgary.ca/People/Norman/encm339fall12005/>

1. Draw a diagram for point one when `low == 4, high == 5`.
2. Draw a diagram for point two when `low == 4, high == 6`.
3. What is the program output?
4. Describe in words
 - (a) what the base case(s) is (are);
 - (b) how it is that the function always makes progress towards a base case;
 - (c) how the function solves a problem by using the solution to a simpler problem of the same kind.
5. Suppose main is changed so that `x` has 1025 elements, with `x[0] < x[1], ..., x[1023] < x[1024], x[2] == 77, x[3] == 79`. Suppose the call to `index_of` is

```
i = index_of(78, x, 0, 1024);
```

What is the program output? How many times is the comparison `key <= a[mid]` made? How many times is the comparison `key == a[low]` made?

Remarks:

- `index_of` is a recursive implementation of a well-known algorithm called *binary search*. Binary search is a much faster way to search a large array than *linear search*, which simply traverses the array looking for a match. (A large array would be one with a lot more than 7 elements!) However, binary search requires a sorted array.
- Generating output in a search function would normally be unhelpful. `cout` is used in this exercise to make you think about the order in which events happen.

```
#include <iostream>
#include <cassert>

int index_of(int key, const int *a, int low, int high);
// REQUIRES: low >= 0 && low <= high
// Elements a[low] ... a[high] exist, and if low < high, then
// a[low] < a[low + 1], ..., a[high - 1] < a[high].
// PROMISES: If key == a[k] for some k with k >= 0 and k <= high,
// returns k. Otherwise returns -1.

int main()
{
    int x[] = { 10, 21, 33, 41, 53, 62, 74 };
    int i;
    i = index_of(53, x, 0, 6);
    return 0;
}

int index_of(int key, const int *a, int low, int high)
{
    assert(low >= 0 && low <= high);
    std::cout << "low == " << low << "; high == " << high << ".\n";
    int result, mid;

    // point one

    if (low == high) {
        if (key == a[low])
            result = low;
        else
            result = -1;
    }
    else {
        mid = (low + high) / 2;
        if (key <= a[mid])
            result = index_of(key, a, low, mid);
        else
            result = index_of(key, a, mid + 1, high);
    }
    std::cout << "low == " << low << "; high == " << high
        << "; result == " << result << ".\n";

    // point two

    return result;
}
```