**Name: ___________________________**

# CS 0449: Lab 2b – Pointers and Recursion

## Binary Search

We've learned several divide-and-conquer type algorithms that are amenable to recursive implementations. One of them was binary search, where a value in a sorted array is searched using the following algorithm expressed in pseudo-code:

```
binary_search(array, value)
{
  middle = <Get value at the bisection of array>
  if <middle is equal to value> then <value found>
  if <middle is larger than value> then binary_search(<left side of array>, value)
  if <middle is less than value> then binary_search(<right side of array>, value)
}
```

Given the following array and the search value 32,

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Array:** | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |

The light gray boxes demarcate the portion of the array under consideration at each step, and the dark gray boxes show the "middle" element at each step.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Step 1** | ░ 1 | ░ 2 | ░ 4 | ░ 8 | ▓ 16 | ░ 32 | ░ 64 | ░ 128 | ░ 256 | ░ 512 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Step 2** | 1 | 2 | 4 | 8 | 16 | ░ 32 | ░ 64 | ▓ 128 | ░ 256 | ░ 512 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Step 3** | 1 | 2 | 4 | 8 | 16 | ▓ 32 | ░ 64 | 128 | 256 | 512 |

Note the self-referential, or recursive, nature of the algorithm (each step is a repetition of the previous step). The algorithm leverages the fact that the array is sorted to find the value in $\log_2 N$ steps, rather than N steps required in a naïve sequential search algorithm.

**What you need to do**

Your task is to write a program which:

- Takes the searched for value (or the needle) as a command line argument.
- Generate an integer array of length 10 that contains the powers of 2 starting from 1 (identical to the array shown in the above example).
- Implement a recursive function with the following prototype:
  `int* search(int* begin, int* end, int needle);`
- Use the above recursive function to search for the needle value, and if found, print the index in the array where the value is found.
- Use pointer arithmetic to implement the above function. You can use the subscript operator (the [] operator) as an alternative but you will find using pointer arithmetic is more expressive and intuitive.

Given below is skeleton code that already implements part of the functionality. Your job is to fill in the code inside the search function. Feel free to reuse the code. No need to replicate comments.

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

int* search(int* begin, int* end, int needle);

int main(int argc, char **argv)
{
  int num;
  int nums[10], i;
  int *found = NULL;

  if(argc != 2) {
    printf("Usage: search <number>\n");
    return 1;
  }
  num = atoi(argv[1]);
  for(i = 0; i < 10; i++) {
    nums[i] = 1 << i;
  }
  found = search(nums, &nums[9], num);
  if(found) {
    printf("Number %d found in index %d.\n", num, found - nums);
  }
  else {
    printf("Number %d was not found.\n", num);
  }
  return 0;
}

// [Arguments]
// begin: pointer to the beginning of array (under consideration)
// end: pointer to the end of array (under consideration)
// needle: value that is being searched for
// [Return Value]
// If not found: NULL
// If found: pointer to element of array containing needle value

int* search(int* begin, int* end, int needle)
{
  // Implement properly
  return NULL;
}
```

Run your program like the following (assuming you name the source code “search.c”):

```
thoth $ gcc search.c –o search
thoth $ ./search 32
Number 32 found in index 5.
thoth $ ./search 64
Number 32 found in index 6.
thoth $ ./search 50
Number 50 was not found.
```

Show the above result to your TA and hand in your results.

## What to Hand In

Assuming you are working under the directory lab2b/,

```
cd ..
tar cvf USERNAME_lab2b.tar lab2b
gzip USERNAME_lab2b.tar
cp USERNAME_lab2b.tar.gz ~wahn/submit/449/RECITATION_CLASS_NUMBER
```