Lab # 15

January 12, 2017

Task 1: Modify the following program to use a recursive `binarySearch` function to perform the binary search of the array. The function should receive an integer array, the starting subscript, the ending subscript and the search key as arguments. If the search key is found, return the array subscript; otherwise, return –1. **(5 marks)**

```
1  // Fig. 6.19: fig06_19.c
2  // Binary search of a sorted array.
3  #include <stdio.h>
4  #define SIZE 15
5
6  // function prototypes
7  size_t binarySearch(const int b[], int searchKey, size_t low, size_t high);
8  void printHeader( void );
9  void printRow( const int b[], size_t low, size_t mid, size_t high );
10
11 // function main begins program execution
12 int main( void )
13 {
14    int a[ SIZE ]; // create array a
15    size_t i; // counter for initializing elements of array a
16    int key; // value to locate in array a
17    size_t result; // variable to hold location of key or -1
18
19    // create data
20    for ( i = 0; i < SIZE; ++i ) {
21       a[ i ] = 2 * i;
22    } // end for
23
24    printf( "%s", "Enter a number between 0 and 28: " );
25    scanf( "%d", &key );
26
27    printHeader();
28
29    // search for key in array a
30    result = binarySearch( a, key, 0, SIZE - 1 );
31
32    // display results
33    if ( result != -1 ) {
34       printf( "\n%d found in array element %d\n", key, result );
35    } // end if
36    else {
37       printf( "\n%d not found\n", key );
38    } // end else
39 } // end main
40
```

```c
// function to perform binary search of an array
size_t binarySearch(const int b[], int searchKey, size_t low, size_t high)
{
   int middle; // variable to hold middle element of array

   // loop until low subscript is greater than high subscript
   while ( low <= high ) {

      // determine middle element of subarray being searched
      middle = ( low + high ) / 2;

      // display subarray used in this loop iteration
      printRow( b, low, middle, high );

      // if searchKey matched middle element, return middle
      if ( searchKey == b[ middle ] ) {
         return middle;
      } // end if

      // if searchKey less than middle element, set new high
      else if ( searchKey < b[ middle ] ) {
         high = middle - 1; // search low end of array
      } // end else if

      // if searchKey greater than middle element, set new low
      else {
         low = middle + 1; // search high end of array
      } // end else
   } // end while

   return -1; // searchKey not found
} // end function binarySearch

// Print a header for the output
void printHeader( void )
{
   unsigned int i; // counter

   puts( "\nSubscripts:" );

   // output column head
   for ( i = 0; i < SIZE; ++i ) {
      printf( "%3u ", i );
   } // end for

   puts( "" ); // start new line of output

   // output line of - characters
   for ( i = 1; i <= 4 * SIZE; ++i ) {
      printf( "%s", "-" );
   } // end for

   puts( "" ); // start new line of output
} // end function printHeader

```

```
96  // Print one row of output showing the current
97  // part of the array being processed.
98  void printRow( const int b[], size_t low, size_t mid, size_t high )
99  {
100    size_t i; // counter for iterating through array b
101
102    // loop through entire array
103    for ( i = 0; i < SIZE; ++i ) {
104
105       // display spaces if outside current subarray range
106       if ( i < low || i > high ) {
107          printf( "%s", "    " );
108       } // end if
109       else if ( i == mid ) { // display middle element
110          printf( "%3d*", b[ i ] ); // mark middle value
111       } // end else if
112       else { // display other elements in subarray
113          printf( "%3d ", b[ i ] );
114       } // end else
115    } // end for
116
117    puts( "" ); // start new line of output
118 } // end function printRow
```

```
Enter a number between 0 and 28: 25

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
-------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
                              16  18  20  22* 24  26  28
                                          24  26* 28
                                          24*

25 not found
```

```
Enter a number between 0 and 28: 8

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
-------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12
                  8  10* 12
                  8*

8 found in array element 4
```

```
Enter a number between 0 and 28: 6

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
-------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12

6 found in array element 3
```

Task 2: Use a double-subscripted array to solve the following problem. A company has four salespeople (1 to 4) who sell five different products (1 to 5). Once a day, each salesperson passes in a slip for each different type of product sold. Each slip contains: **(5 marks)**

a) The salesperson number

b) The product number

c) The total dollar value of that product sold that day

Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last month is available. Write a program that will read all this information for last month's sales and summarize the total sales by salesperson by product. All totals should be stored in the double-subscripted array `sales`. After processing all the information for last month, print the results in tabular format with each column representing a particular salesperson and each row representing a particular product. Cross total each row to get the total sales of each product for last month; cross total each column to get the total sales by salesperson for last month. Your tabular printout should include these cross totals to the right of the totaled rows and to the bottom of the totaled columns.

Grading and LMS Submission

- Make sure that the lab engineer has graded your programs until 5 pm.
- You've uploaded the C source files in Zip format over LMS until 5:30 pm.