|  |  |
| --- | --- |
| Result = | Puzzle(5) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**p. 36**

**Homework: (5 points) Draw a recursion diagram, and the result of calling the following recursive function with n=5.**

|  |  |
| --- | --- |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| return (8+ | Puzzle(8/2) ) |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| return (4+ | Puzzle(4/2) ) |
| return (8+ | Puzzle(8/2) ) |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| return (2+ | Puzzle(2/2) ) |
| return (4+ | Puzzle(4/2) ) |
| return (8+ | Puzzle(8/2) ) |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |
|  |  |

|  |  |
| --- | --- |
| return (1) |  |
| return (2+ | Puzzle(2/2) ) |
| return (4+ | Puzzle(4/2) ) |
| return (8+ | Puzzle(8/2) ) |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |

|  |  |
| --- | --- |
|  |  |
| return (2+ | 1) |
| return (4+ | Puzzle(4/2) ) |
| return (8+ | Puzzle(8/2) ) |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
| return (4+ | 3 ) |
| return (8+ | Puzzle(8/2) ) |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| return (8+ | 7) |
| return (16+ | Puzzle(16/2) ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| return (16+ | 15 ) |
| return (1\* | Puzzle(3\*5+1) |
| Result = | Puzzle(5) |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| return (1\* | 31 |
| Result = | Puzzle(5) |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Result = | 31 |
|  |  |

**p. 37 Homework (10 points)**

Assume that we will find 25, in list {5, 10, 15, 25, 45}

|  |  |
| --- | --- |
| Result = | bs(list, 0, 4, 25) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| return | bs(list, 3, 4, 25) |
| Result = | bs(list, 0, 4, 25) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Return | bs(list, 3, 3, 25) |
| return | bs(list, 3, 4, 25) |
| Result = | bs(list, 0, 4, 25) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Return | 0 |
| Return | bs(list, 3, 3, 25) |
| return | bs(list, 3, 4, 25) |
| Result = | bs(list, 0, 4, 25) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
| Return | 0 |
| return | bs(list, 3, 4, 25) |
| Result = | bs(list, 0, 4, 25) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
| return | 0 |
| Result = | bs(list, 0, 4, 25) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| Result = | 0 |
|  |  |
|  |  |
|  |  |
|  |  |

**State what the program is designed to do.**

It calls ‘binary search’ using the recursion.

This function wants to find the index of the key value.

‘list[]’ is the array. ‘lo’ and ‘hi’ are list’s index. (lo is 0 and hi is ‘the length of list -1’)

Key is the number in the ‘list[]’.

And ‘mid’ is the middle of the value ‘lo’ and ‘hi’.

if (lo > hi)

return -1; //this is wrong ‘lo’ and ‘hi’, so return -1.

mid = (lo + hi) / 2; //calculate the ‘mid’ using ‘lo’ and ‘hi’

if (list[mid] == key)

return 0; //if we find the ‘key’, then return 0

else if (list[mid] > key) //when the value of list[mid] is higher than key

bs(list, lo, mid-1, key); //use ‘bs’function one more time but change the value of ‘hi’ to ‘mid-1’

else if (list[mid] < key) ////when key is higher than the value of list[mid]

bs(list, mid+1, hi, key); //use ‘bs’function one more time but change the value of ‘lo’ to ‘mid+1’

In the recursion function, we have to change the value (lo and hi).

The value of ‘lo’ should be increased by 1 and the value of ‘hi’ should be reduced by 1.

If the return’s value is 0, then that mid’s value is the key’s index.

* list[mid] = key

!! Find the value of key using the list’s index and middle value of list.