CMPS 12B 3-14-11

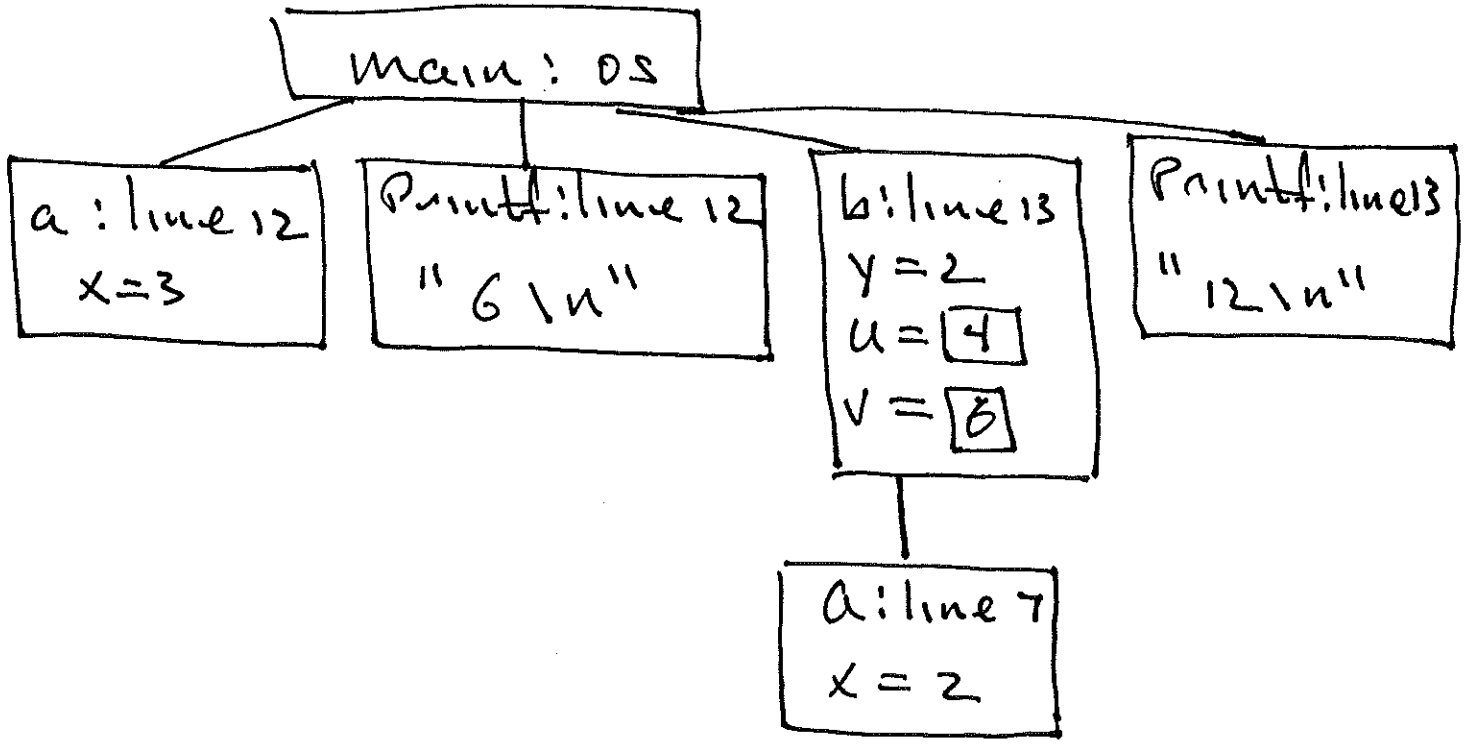
- Final Tue 12-3
- asg5 : no extension.
- Today fcn call stack

  stack Trace of C Program

Ex.

```
1   #include<stdio.h>
2   int a(int x){
3       return 2*x;
4   }
5   int b(int y){
6       int u,v;
7       u=a(y);
8       v=u+y;
9       return 2*v;
10  }
11  int main(void){
12      Printf("%d\n", a(3));
13      Printf("%d\n", b(2));
14      return 0;
15  }
```

'Recursion' tree:



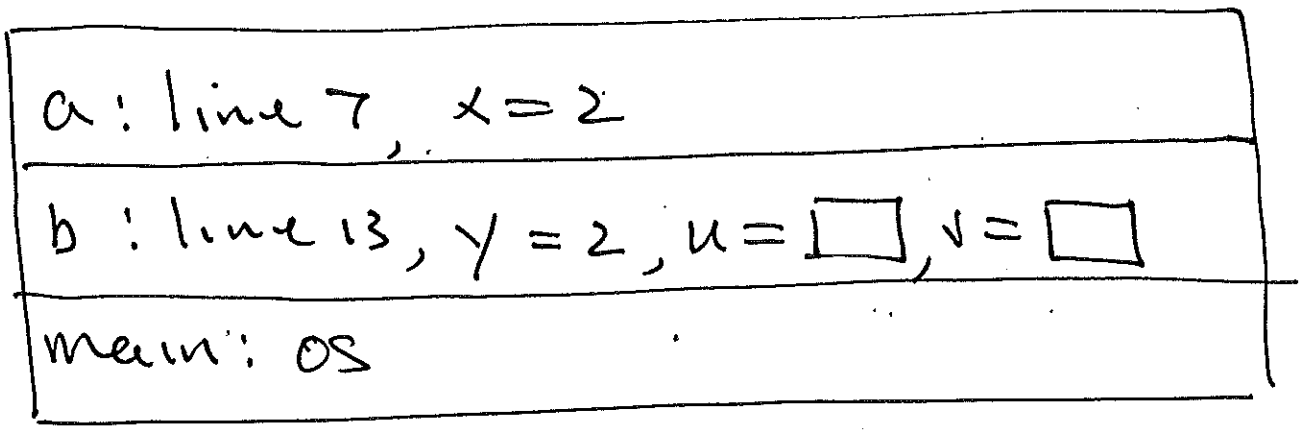
trace of fcn call stack:

(empty)	main	a main	main	Printf main	main	b main

a b main	b main	main	Printf main	main	(empty)

*

Snapshot of stack at Point *



Quicksort:

See notes at:

http://www.soe.ucsc.edu/classes/CMPS0126/Winter09/notes.html

Pages: 111 - 113

Also see Program Sort.c in

.../Winter09/Lecture

```
void QuickSort(int * A, p, r) {
    int q;
    if (p < r) {
        q = Partition(A, p, r);
        QuickSort(A, p, q-1);
        QuickSort(A, q+1, r);
    }
}
```

PARTITION REARRANGES THE SUBARRAY $A[p \cdots r]$ AND RETURNS AN INDEX q SUCH THAT

$$A[p \cdots (q-1)] \leq A[q] < A[(q+1) \cdots r]$$

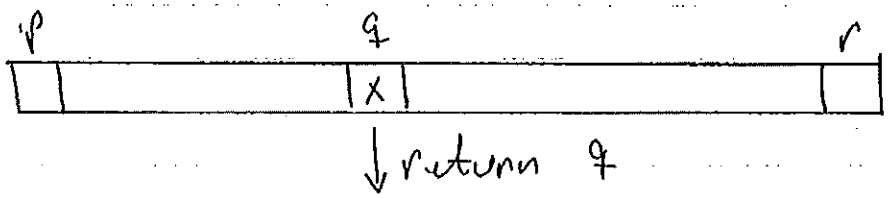
THE ELEMENT $A[q]$ IS CALLED THE PIVOT.

```
int Partition(A, p, r) {
    int i, j, x;
    x = A[r];
    i = p-1;
    for (j = p; j < r; j++) {
        if (A[j] <= x) {
            i++;
            swap(A, i, j);
        }
    }
    swap(A, i+1, r);
    return(i+1);
}
```

WE USE A HELPER FUNCTION SWAP WHICH EXCHANGES ARRAY ELEMENTS.

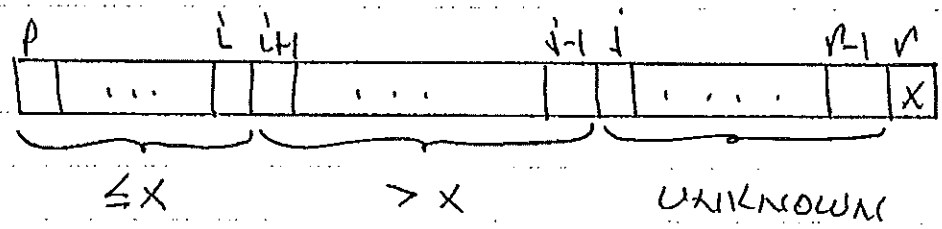
```
void swap(int* A, int i, int j){
    int temp;
    temp = A[i];
    A[i] = A[j];
    A[j] = temp;
}
```

PARTITION PICKS AN ELEMENT IN A[p...r] TO BE THE SO-CALLED PIVOT. IN OUR CODE THE PIVOT IS $x = A[r]$. (IN THE BOOK IT A[p].) PARTITION THEN DIVIDES THE SUBARRAY A[p..r] INTO TWO SECTIONS: THOSE ELEMENTS LESS THAN OR EQUAL TO x AND THOSE WHICH ARE GREATER THAN x. IT THEN SWINGS THE PIVOT INTO A POSITION BETWEEN THE TWO SECTIONS, THEN RETURNS THE INDEX OF THE PIVOT.

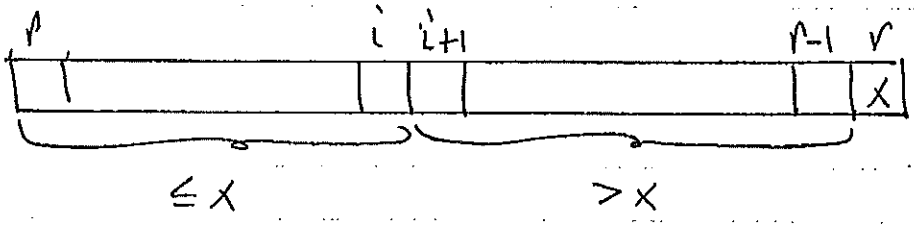


THUS $A[p \ldots q-1] \leq A[q] < A[q+1 \ldots r]$ UPON RETURN OF PARTITION. ALL THAT IS LEFT IS TO SORT SUBARRAYS $A[p \ldots q-1]$ AND $A[q+1 \ldots r]$ RECURSIVELY, WHICH QUICKSORT DOES.

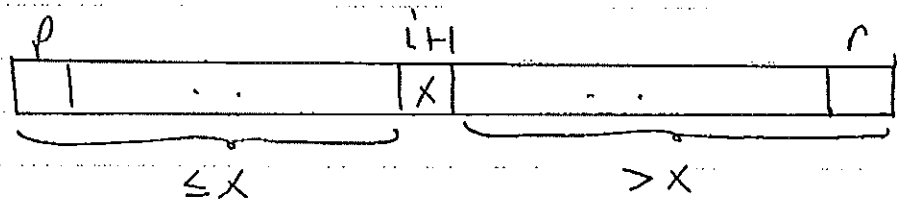
EXAMINING THE CODE FOR PARTITION WE SEE THAT THE FOR LOOP MAINTAINS THE INVARIANT



i.e. $A[p \ldots i] \leq x < A[i+1 \ldots j-1]$. THIS INVARIANT IS CERTAINLY TRUE BEFORE THE FIRST ITERATION, SINCE BOTH SUBSECTIONS ARE EMPTY. IT'S NOT HARD TO SEE THAT THE INVARIANT IS MAINTAINED FROM ONE ITERATION TO ANOTHER. WHEN $j = r$ WE HAVE



THE SINGLE SWAP $A[i+1] \leftrightarrow A[r]$ GIVES



THEN WE RETURN THE INDEX $i+1$.

QUICK SORT IS DUAL IN SOME SENSE TO MERGE SORT. BOTH ARE DIVIDE AND CONQUER ALGORITHMS. MERGE SORT DOES ITS REAL WORK IN THE RECOMBINE STEP, WHILE QUICKSORT DOES IT IN THE DIVIDE STEP.