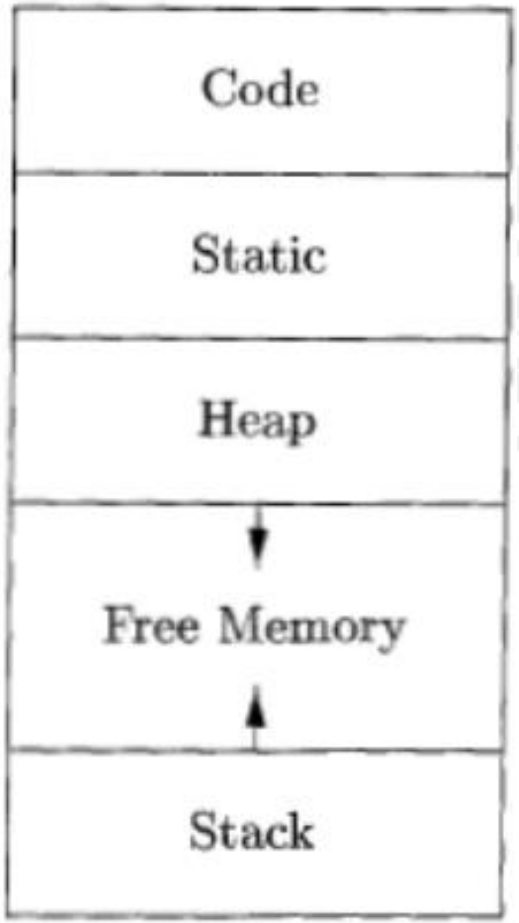
Run Time Environments

Introduction

- Compiler must cooperate with OS.
- Creates and manages the runtime environment .
- Compiler must do the storage allocation and provide access to variables and data.
- Memory management
  - Stack allocation
  - Heap management
  - Garbage collection

Storage organization



Static and dynamic storage allocation

- Static: Compile time
- Dynamic: Runtime allocation
- Many compilers use some combination of following
  - Stack storage: for local variables, parameters and so on
  - Heap storage: Data that may outlive the call to the procedure that created it
- Stack allocation is a valid allocation for procedures since procedure calls are nested

Stack allocation of space

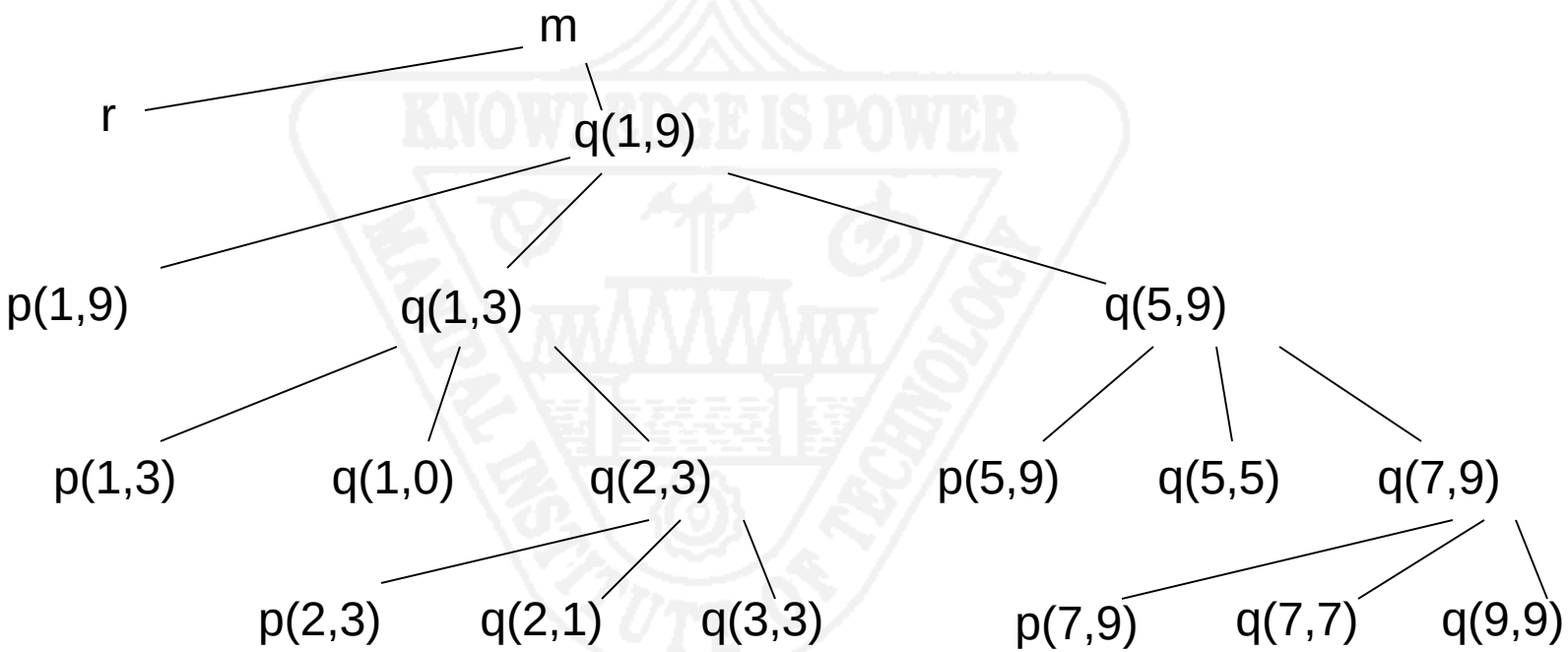
- Each time procedure is called, activation record is pushed.
- Each time procedure is returned, activation record is popped.

```
int a[11];
void readArray() { /*reads 9 ints into a[1]-a[9] */
}

int partition(int m, int n) { /*picks a separator value v, and
   partitions a[m..n] so that a[m..p-1] are less than v,
   a[p]=v and a[p+1,n] are equal to or greater than v.
   Returns p. */
}

void quicksort(int m, int n) {
   int I;
   if (n > m){
        i=partition(m,n);
        quicksort(m,i-1);
        quicksort(i+1,n);
   }
}

main() {
   readArray();
   a[0] =-9999;
   a[10]= 9999;
   quicksort(1,9);
}
```

enter main()
- enter readArray()
- leave readArray()
- enter quicksort()
  - enter partition(1,9)
  - leave partition(1,9)
  - enter quicksort(1,3)
    - …
  - leave quicksort(1,3)
  - enter quicksort(5,9)
    - …
  - leave quicksort(5,9)
- leave quicksort(1,9)

leave main()

Activation Tree



Suppose that control lies within a particular activiation
Of some procedure, N, of the activation tree. The stack
Corresponds to the ancestors of node N.

Three cases nested procedure termination

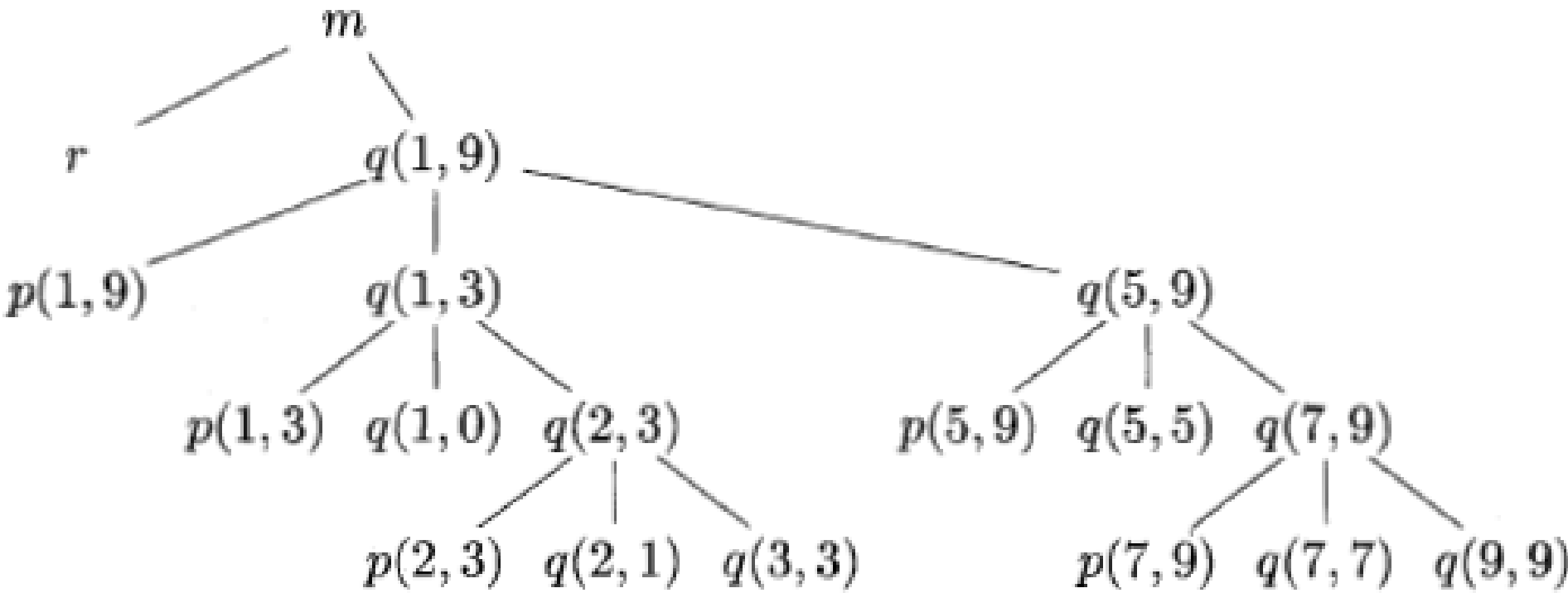
P calls q

1. Activation of q terminates normally. P resumes.
2. Activation of q or some procedure called by q , aborts. P aborts.
3. Activation of q terminates bcz of exception which it cannot handle, but P can handle. P continues, may not be from same point. If P cannot, procedure which called P, will try to handle.

Possible Activations

```
enter main()
    enter readArray()
    leave readArray()
    enter quicksort(1,9)
        enter partition(1,9)
        leave partition(1,9)
        enter quicksort(1,3)
            ...
        leave quicksort(1,3)
        enter quicksort(5,9)
            ...
        leave quicksort(5,9)
    leave quicksort(1,9)
leave main()
```

Activation tree

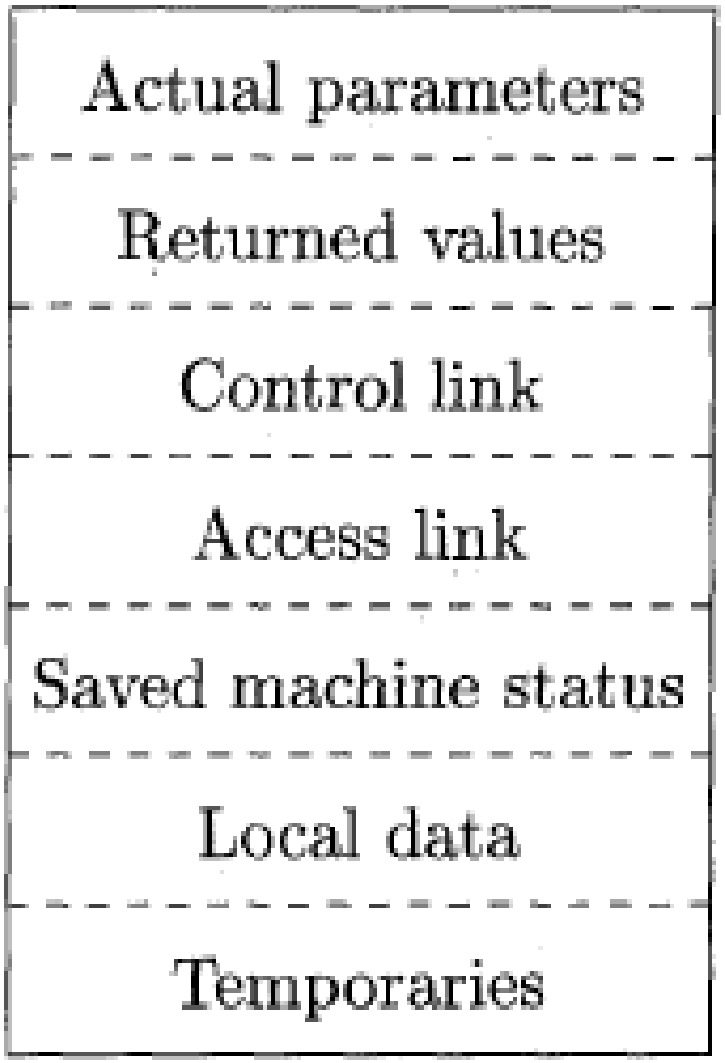


Activation tree[contd..]

- Procedure calls – preorder traversal
- Return calls – postorder traversal
- Node N and its ancestors are alive

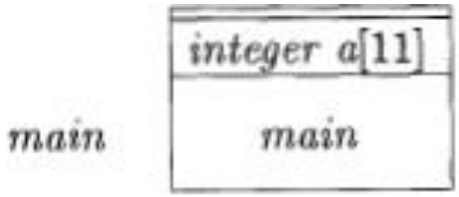
Activation Record

- Calls and returns are managed by control stack.
- Each live activation has an activation record on control stack.

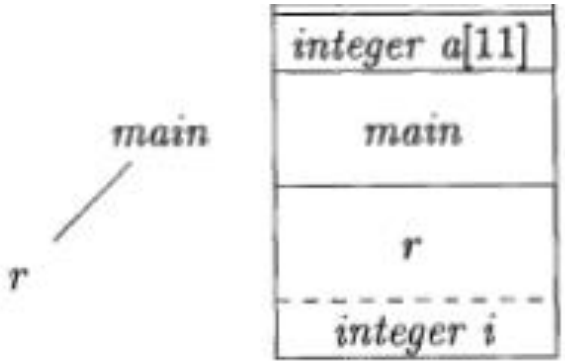


Pts to act rec of caller

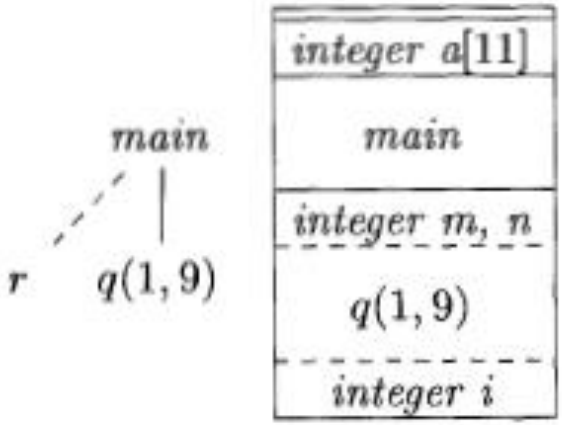
Act rec of some pro called



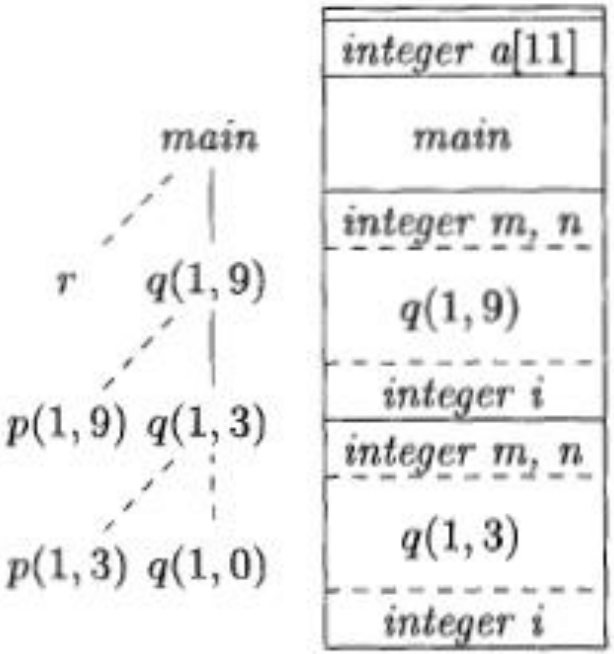
(a) Frame for *main*



(b) r is activated



c) r has been popped and $q(1,9)$ pushed



(d) Control returns to $q(1,3)$