Outcomes

- Stack Allocation of Space
- Activation Trees
- Activation Records

STACK ALLOCATION OF SPACE

- Whenever a procedure is called,
  - **space for its local variables is pushed onto the stack**
- When the procedure terminates,
  - **space is popped off**

- Almost all compilers for languages that use procedures, functions, or methods manage their run-time memory as a stack.

ACTIVATION TREES

- Activation:
  The sequence in which the procedure executes

- Activation Tree:
  Procedures execute sequentially
  This sequence is easily represented by a tree

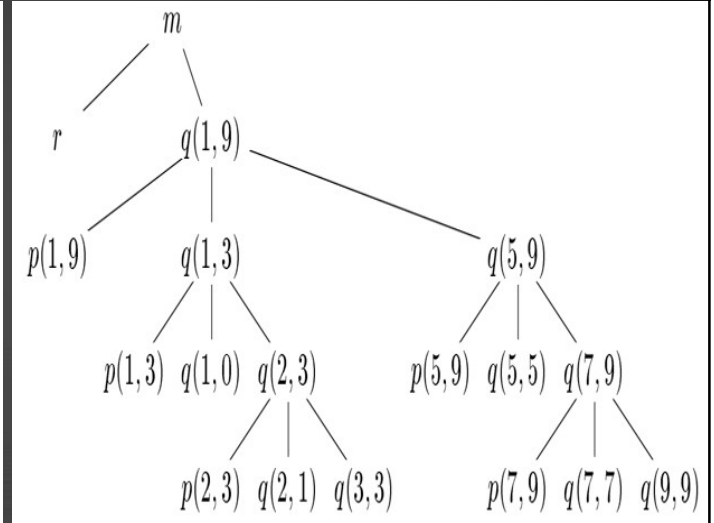
Example: Recursive Quicksort Algorithm

```
int a[11];
void readArray() {
     int i;
     ...
}
int partition(int m, int n) {
     ...
}
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);
}
```

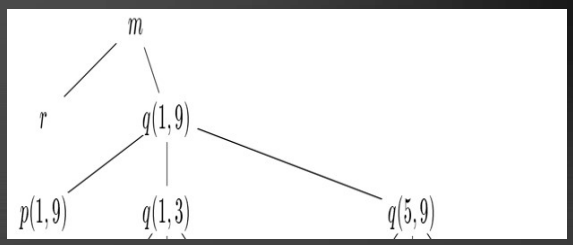
Possible Activations:

```
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:



- In general, procedure activations are nested in time.
- If the activation of procedure p calls procedure q, then that activation of q must end before the activation of p can end.

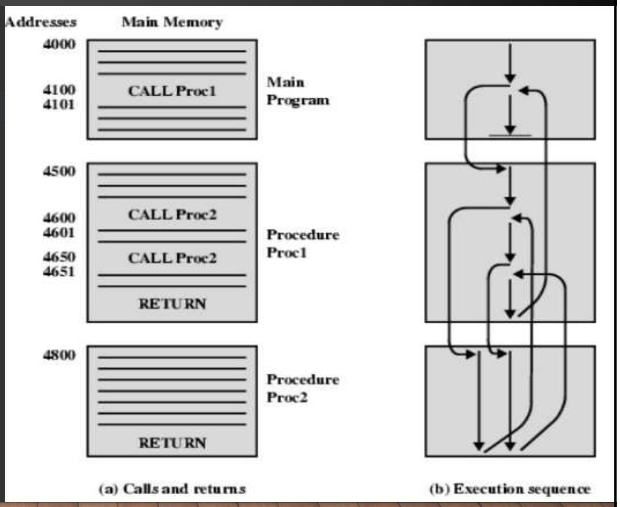


3 COMMON CASES:

- Case 1: Activation of functionA terminates normally.
- Case 2: Activation of functionA aborts.
- Case 3: Main handles the exception of functionA

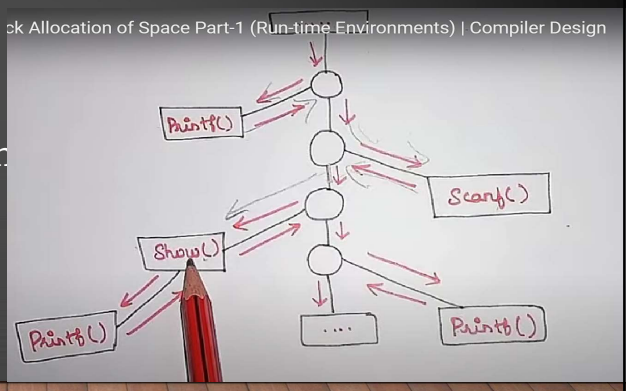
Program Flow

```
1  #include <iostream>
2
3  void functionB() {
4      std::cout << "Inside function B\n";
5  }
6
7  void functionA() {
8      std::cout << "Inside function A\n";
9      functionB();
10 }
11
12 int main() {
13     std::cout << "Inside main function\n";
14     functionA();
15     return 0;
16 }
17
```



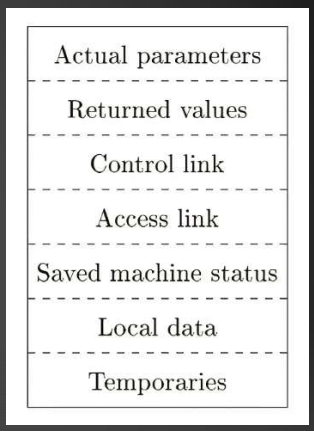
Use of Run-Time Stack

- Sequence of Procedure calls → PreOrder Traversal
- Sequence of Returns → PostOrder Traversal
- The live Activations of node N and its ancestors are in the order of their appearance along the path to N from the root, and they will return in Reverse Order.



Activation Records

- Procedure calls and returns are usually managed by a run-time stack called the control stack.
- Each live activation has an activation record(sometimes called a frame) on the control stack
- The contents of activation records vary with the language being implemented. Here is a list of the kinds of data that might appear in an activation record:



- Temporaries
  Values that arise from the evaluation of expression
- Local Data
  Procedure data is stored locally.
- Saved Machine Status
  The machine status(register, program counter) before the procedure call is saved

- Access Link
  Non-local data information is stored
- Control Link
  Points to the caller's activation record.
- Returned Values
  Stores the return value of the function, if any.
- Actual Parameter
  Stores the actual parameters of the calling procedure.

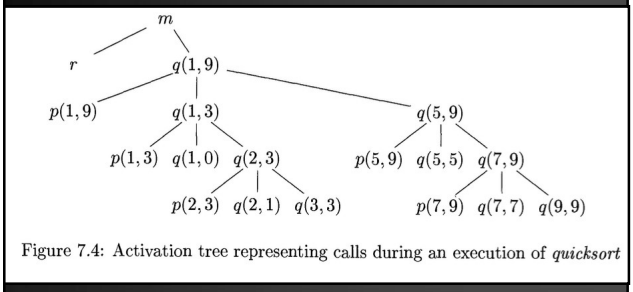


Figure 7.4: Activation tree representing calls during an execution of *quicksort*