① Explain different source language issues with suitable examples

Procedure

It is a declaration that associates with an identifier with an statement. The identifier with a statement. The identifier is a procedure name, and statement is procedure body.

eg:

```
procedure readarray,
 var i: integer,
  begin
  for i:= 1 to 9 do read(a[i])
  end;
```

Activation tree

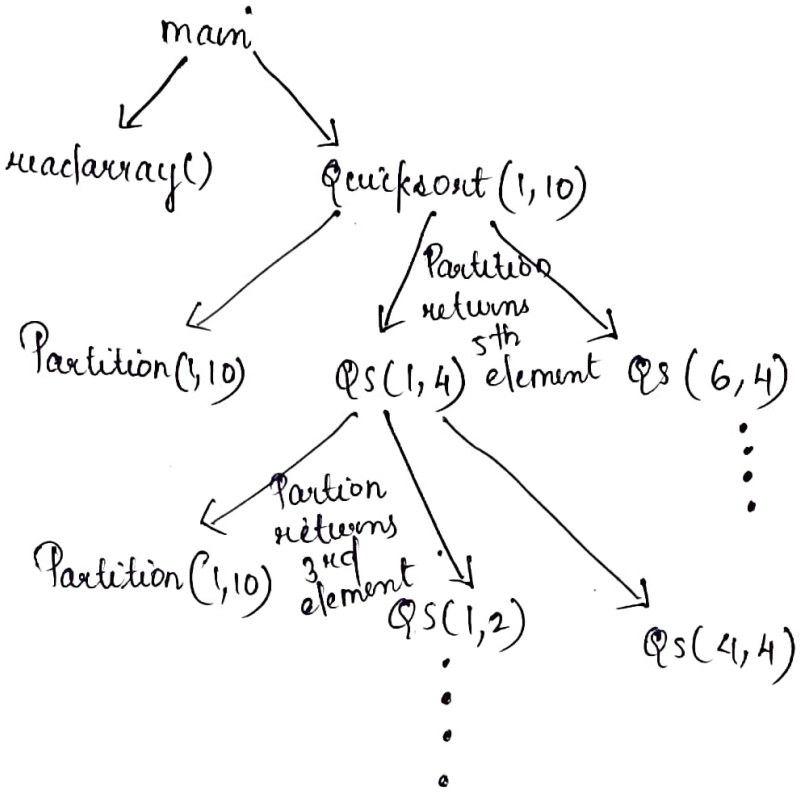
- Each execution of procedure is referred to as an activation of the procedure. Lifetime of an activation is the sequence of step present in the execution of the procedure.
- If 'a' and 'b' be two procedures then their activations will be non-overlapping or nested.
- A procedure is recursive if a new activation begins before an earlier activation of the same procedure has ended.

eg:

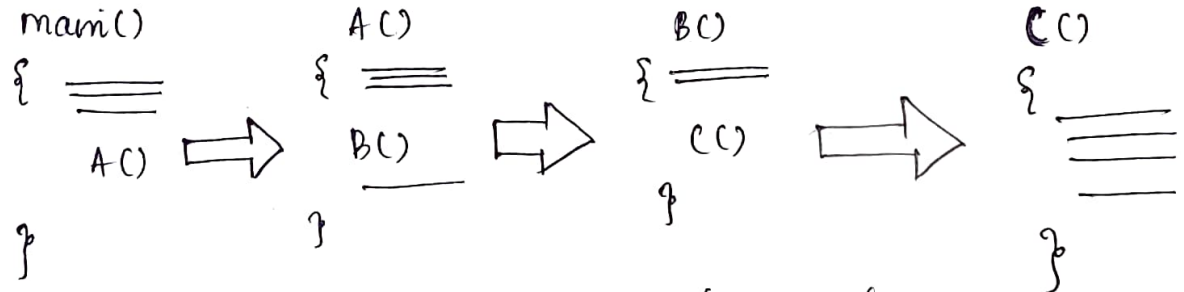
Consider the following program of quicksort

```
main()
  {
    readarray();
    quicksort(1,10);
  }
quicksort(int m, int n)
  {
    int i= partition(m,n)
```

```
    quicksort(m, i-1);
    quicksort(i+1, n);
}
```



- First main function was root the main calls readarray and quicksort.
- Quicksort in turn calls partition and quicksort again. The flow of control in a program corresponds to the depth first traversal of activation tree which starts at root.



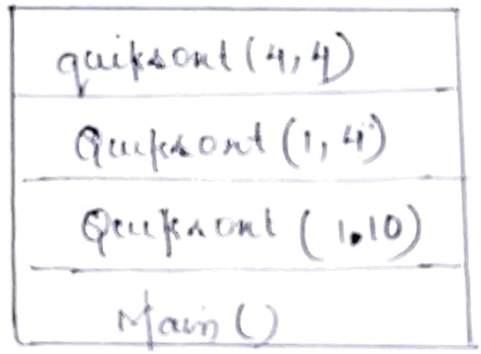
Before going to A() activation record of main() is created when A() is called activation record of A() is created.

Before going to B() activation record of main and A() in stack

Before going to C() activation record of main and B() pushed in stack

Control Stack

- used to keep track of the live procedures activations i.e procedures whose execution have not been completed.
- A procedure name is pushed onto the stack when it is called and it is popped when it returns.
- when a procedure is called, an activation record is pushed onto the stack and as soon as the control returns the caller function the activation record is popped.
- Then the content of the control stack are related to paths to the root of the activation tree. when node n is at the top of the control stack, the stack contains the nodes along the path from n to the root
- Consider the above activation tree, when quicksort (4,4) gets executed, the contents of control stack where main(), quicksort (1,10), quicksort (1, 4) quicksort (4,4)



The Scope of Declaration.

- The declaration is a syntatic construct that associates information with a name.

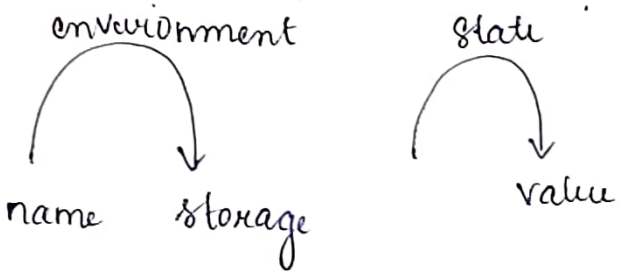
Declaration may be explicit such as

```
var i : integer;
```

or may be implicit.

Binding of Names

- Even all object if each name is declared once in a program, the same name may denote different data object at run time "Data objects" corresponds to a storage location that hold values.
- The term environment refers to a function that maps a name to storage location.
- The term state refers to a function that maps a storage to the value held there.



(2) Explain Machine dependent & Independent optimization.

Machine-dependent optimisation:

Machine-dependent optimization is done after the target code has been generated and when the code is transformed according to the target machine architecture. It involves CPU registers and may have absolute memory references rather than relative references. Machine-dependent optimizers put effort to take maximum advantage of memory hierarchy.

Machine-Independent optimization:

- Machine Independent optimization attempts to improve the intermediate code to get a better target code. The part of code

which is transformed here does not involve any absolute memory location or CPU registers.

- The process of intermediate code generation introduces much inefficiency like: using variable instead of constants, extra copies of variable, repeated evaluation of expression. Through the code optimization, you can remove such efficiencies and improves code.

- It can change the structure of the program sometime of beyond recognition like: unroll loops, inline functions, eliminates some variable that are programmer defined.

③ Explain local and global optimization.

Local optimisation.

The local optimisation is performed within a straightline or basis block of code without any information from any other block. The different local optimizations that can be applied to a procedure are:

1. Constant folding i.e replacing the run-time computations by the compile computations.

2. Common subexpression elimination. i.e the value resulting from the calculation of a subexpression is used multiple times perform the calculation once and substitute the result of each individual calculations.

Global optimization.

The local optimization involves the statements within a basic block. All the other optimization are called global optimization. The global optimization allows the compiler to look out the overall program and determine how best to apply the desired optimisation level. The global optimization is generally performed by using data flow analysis the transmission of used relationship from all parts of the program to the places where the information can be of use.