Yash Mahajan SE IT B 191061
Roll No - 04 Subject - PCPF

Assignment Test 1

Q.1 (a) Define local and global variable wrt imperative programming paradigm.

Local variable :-
It is a variable declared within programming block or subroutine. It can only be used inside the subroutine or code block in which it is declared. The variable exists only until the block of the function is under execution.

Global variable :-
A global variable in the program is a variable defined outside the subroutine or function. It has a global scope means it holds its value throughout the lifetime of the program. Hence, it can be accesed throughout the program.

Eg :-

```
int g;          // global variable
int main (){
int a = 5 ;     // Local variable
g = 7 ;
```

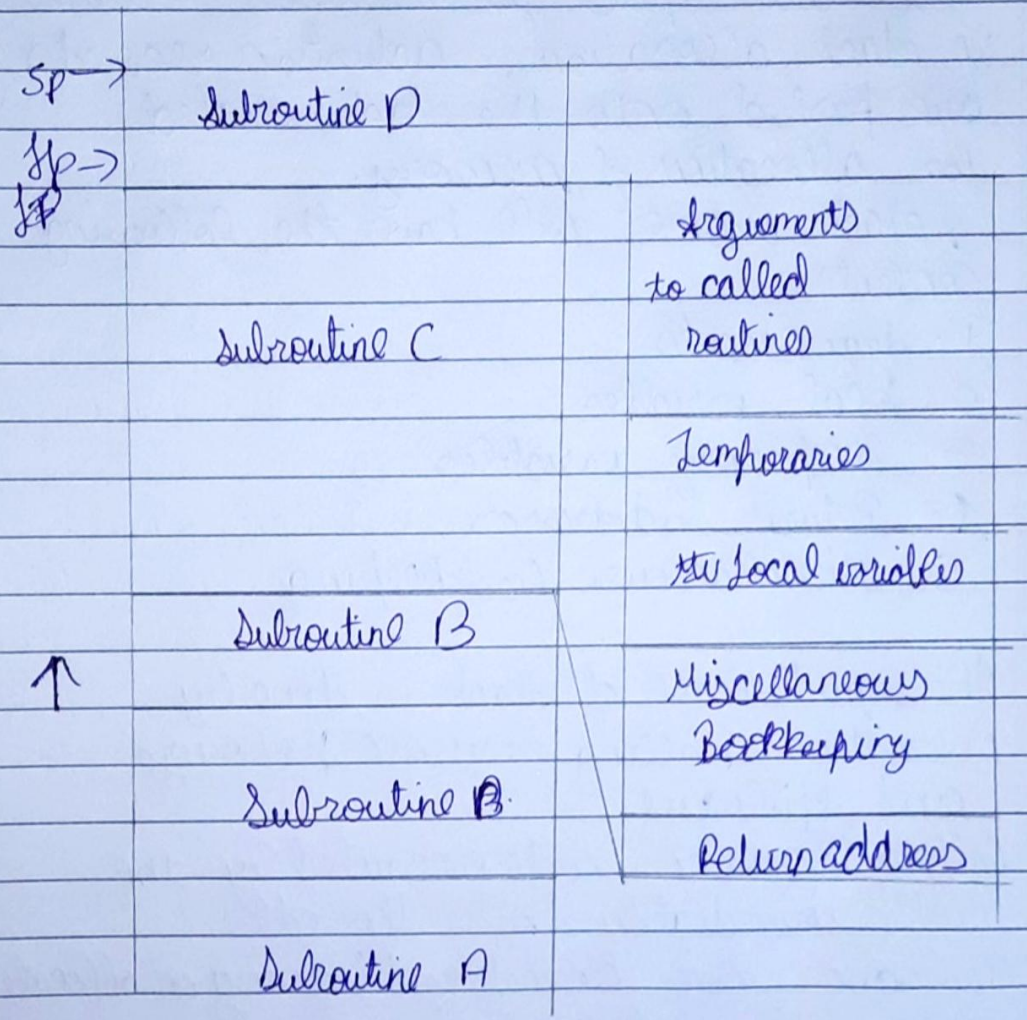
```
return a * &g;
}
```

Q.2

(b) Differentiate between static and dynamic scoping.

Static Scoping	Dynamic Scoping
In static scoping (or lexical scoping), definition of a variable is resolved by searching its containing block or function, if it fails it then searching in outer block and so on.	In dynamic scoping definition of a variable is resolved by searching its containing block or function if that fails then searching its calling function, if not found the it searches in the function that called the calling function.
Most modern programming languages support static scoping.	Dynamic scoping is supported by older languages
Eg Algol, C, Pascal	Eg. SNOBOL, APL
In static scoping we can associate scope of variable at compile time	In dynamic scoping we can associate the scope of variable at run time

Q.2. Explain stack allocation with help of neat labelled diagram and example.



Procedure C -- main program
D, E A.
procedure B
if .. then B else C
Procedure A
B

for recursion static allocation is not possible
Number of instances of variable is unbounded
stack allocation allows us to allocate space for recursive routines.
In stack allocation, activation records are pushed onto the stack created for allocation of memory.
A stack frame will have the following contents :-

1. Arguements
2. Local variables
3. Temporary variables
4. Return addresses
5. Miscellaneous bookkeeping

The maintainence of stack is done by subroutine calling sequence, prologue and epilogue.
Calling Sequence : Code executed by the caller immediately after the call.
Prologue : Code executed at begining of, allocates a frame by substracting frame from sp.
Saves callee - saves registers used anywhere inside callee.
Epilogue : - Code executed at the end and puts return values into registers, restores saved registers using spaceset adds sp to de-allocate frame.