# 1.

Which of the above three functions are likely to cause problems with pointers? (GATE 2001)

A Only P3

B Only P1 and P3

**C Only P1 and P2**

D P1, P2 and P3

In P1, pointer variable x is a local variable to g(), and g() returns pointer to this variable. x may vanish after g() has returned as x exists on stack. So, &x may become invalid.

In P2, pointer variable px is being assigned a value without allocating memory to it.

P3 works perfectly fine. Memory is allocated to pointer variable px using malloc(). So, px exists on heap, it’s existence will remain in memory even after return of g() as it is on heap.

# 2.

# include<stdio.h>

# include<stdlib.h>

void fun(int \*a) {

a = (int\*)malloc(sizeof(int));

}

int main() {

int \*p;

fun(p);

\*p = 6;

printf("%d\n",\*p);

return(0);

}

**A May not work**

B Works and prints 6

The program is not valid. Try replacing “int \*p;” with “int \*p = NULL;” and it will try to dereference a null pointer.

This is because fun() makes a copy of the pointer, so when malloc() is called, it is setting the copied pointer to the memory location, not p.

p is pointing to random memory before and after the call to fun(), and when you dereference it, it will crash.

If you want to add memory to a pointer from a function, you need to pass the address of the pointer (ie. double pointer).

# 3.

which languages necessarily need heap allocation in the run time environment?

A Those that support recursion

B Those that use dynamic scoping

C Those that use global variables

**D Those that allow dynamic data structures**

# 4.

What happens when delete is used for a NULL pointer?

int \*ptr = NULL;

delete ptr;

A Compiler Error

B Run-time Crash

**C No Effect**

Deleting a null pointer has no effect, so it is not necessary to check for a null pointer before calling delete.

# 5.

Is it fine to call delete twice for a pointer?

#include<iostream>

using namespace std;

int main() {

int \*ptr = new int;

delete ptr;

delete ptr;

return 0;

}

A Yes

**B No**

It is undefined behaviour to call delete twice on a pointer. Anything can happen, the program may crash or produce nothing.