Return value (fork)

Program A	Output rare			Program B	

int q;
printf("%d %d\n",getpid( ),getppid( ));
q=fork( );
printf("%d%d%d\n",getpid(),getppid(),q);
	AL
	ALB
	B10
		AL
		BA0
		ALB
			AL
			ALB
			BA0
				int q;q=fork( ); if (q==0) fork( );
				printf("%d%d\n",getpid( ),getppid( ));
				sleep(1); o/p AL BA CB
				(the order may be different)	

The system call fork( ) is a function with side effect. As the side effect, two copies of the running process are made. The function call fork( ) returns 0 in the child process and the child's id in the parent process.
The second program creates process B and C. The parent of B is A and the parent of C is B.

Program 1	Program 2

int p=fork( );
int q=fork( );
if (p==0) fork( );
fork( );
printf("X");
	int p,q;p=fork( );
	fork( );q=fork( );
	if (p==q)
	printf("X");
	else printf("Y");

1. How many X will be printed by program 1?
2. How many X and Y are printed by the program 2?
3. Write a program to create 10 additional processes. Every process prints its id. Use only 4 fork's. No getpid, getppid. (except in print)
4. Write a program, which reads (int n) and creates n additional processes. A process can use atmost $(\log_2 n)+1$ fork's.
5. Write a program, which creates processes B, C, D, and E. The parent of B, C and D is A. The parent of E is B.
6. Read n and create a system of 2^n-1 processes. Every process has either two children or no child.
   Let printf("[%d %d]",getpid( ),getppid( )); sleep(1); is written at the end
   o/p for n=3 [AL],[BA],[CA],[DB],[EB],[FC],[GC]

```
int aork( )
{   int p;
    p=fork( );
    if (p==0)
       return(getppid( ));
    else return(p);
}
main( )
{   int q;
    q=aork( );
    printf("%d",getpid( ));
    printf("%d",getppid( ));
    printf("%d\n",q);
    sleep(1);
}
```

A function aork( ) is defined. It creates one additional process. In the parent the id of child is returned. In child the id of parent is returned. Output ALB BAA
If we remove sleep(1) then parent of B may be outputted as 1.

1. Define a function bork( ). It creates two additional processes. Let A be the id of parent and B and C are id's of the child. The parent of both B and C is A. The function returns C in parent process, B in C, and A in B. Hence the same main program (when function call q=bork( ) is made) will output (lines may be exchanged) ALC BAA CAB
2. Define a function cork( ). It creates 2 addition processes. Let parent be A, child be B and grandchild be C. A returns id of B. B of C and C of A.
3. Define a function cark( ). It creates 3 addition processes. Let parent be A, child be B, grandchild be C, and great grandchild be D. A returns id of B. B returns id of C, C returns id of D and D returns id of A.

4. Define a function dork(int n). It creates 2^n-1 more processes by using n fork( ). The parent returns 1. Every child (these shall be n children) returns 2. Every grand child returns 3. Those processes, whose father's father 's father's … (n times) ... father is the original process, returns n+1.
5. Define a function eork( ). It creates 10 more processes. Let A be the id of main process and B, C, D, … ,K are id's of other processes. A is father of B, B is father of C, C is father of D, …, J is father of K. In C the function call returns A, in D it returns B, … , in K it returns I, and in A and B it returns 0.

```
#include<stdio.h>
main()
{  printf("%d%d\n",getpid(),getppid());
   fork() && fork();
   printf("%d%d\n",getpid(),getppid());
```

output xy xy zx wx (separate line). Last three outputs in any order.
When fork() && fork() is replaced by

fork( ) || fork( )	xy xy zx wz
(fork( ) && fork( )) || fork( )	xy xy zx wx sw tz
fork( )&&(fork( ) || fork( ))	xy xy zx wx sw

sleep(1); }	

main(){ int p,q; p=fork();q=fork();printf("%d %d %d %d %d\n",p,q,getpid(),getpgrp(),getppid());sleep(1);}

CBAAW 0DCAA C0BAA 00DAC On different execution from window the value of W is unchanged