Q1.

#include <stdio.h>

#include <unistd.h>

int main()

{

if (fork() || fork())

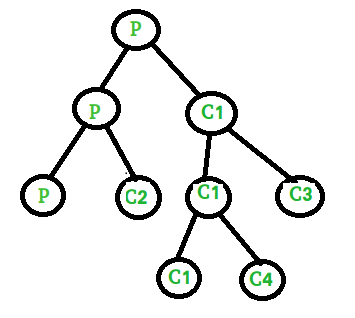
fork();

printf("1 ");

return 0;

}

Output: 1 1 1 1 1



Explanation:

1. It will create two process one parent P (has process ID of child process)and other is child C1 (process ID = 0).  
   2. In if statement we used OR operator( || ) and in this case second condition is evaluated when first condition is false.  
   3. Parent process P will return positive integer so it directly execute statement and create two more processes (one parent P and other is child C2). Child process C1 will return 0 so it checks for second condition and second condition again create two more processes(one parent C1 and other is child C3).  
   4. C1 return positive integer so it will further create two more processes (one parent C1 and other is child C4). Child C3 return 0 so it will directly print 1.

Q 2:

#include <stdio.h>

#include <unistd.h>

int main()

{

if (fork()) {

if (!fork()) {

fork();

printf("1 ");

}

else {

printf("2 ");

}

}

else {

printf("3 ");

}

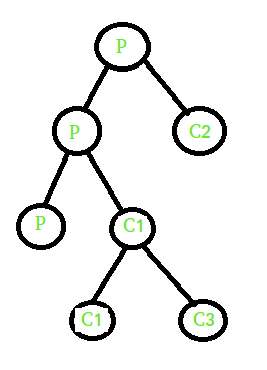
printf("4 ");

return 0;

}

Output:

2 4 1 4 1 4 3 4

  
Explanation:  
1. It will create two process one parent P (has process ID of child process) and other is child C1 (process ID = 0).  
2. When condition is true parent P executes if statement and child C1 executes else statement and print 3. Parent P again check if statement and create two process (one parent P and child C2). In if statement we are using not operator (i.e, !), it executes for child process C2 and parent P executes else part and print value 2. Child C2  
further creates two new processes (one parent C2 and other is child C3).

Q3:

Using fork() to produce 1 parent and its 3 child processes

Program to create four processes (1 parent and 3 children) where they terminates in a sequence as follows :

(a) Parent process terminates at last

(b) First child terminates before parent and after second child.

(c) Second child terminates after last and before first child.

(d) Third child terminates first.

**Solution:**

// CPP code to create three child

// process of a parent

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

// Driver code

int main()

{

int pid, pid1, pid2;

// variable pid will store the

// value returned from fork() system call

pid = fork();

// If fork() returns zero then it

// means it is child process.

if (pid == 0) {

// First child needs to be printed

// later hence this process is made

// to sleep for 3 seconds.

sleep(3);

// This is first child process

// getpid() gives the process

// id and getppid() gives the

// parent id of that process.

printf("child[1] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

else {

pid1 = fork();

if (pid1 == 0) {

sleep(2);

printf("child[2] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

else {

pid2 = fork();

if (pid2 == 0) {

// This is third child which is

// needed to be printed first.

printf("child[3] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

// If value returned from fork()

// in not zero and >0 that means

// this is parent process.

else {

// This is asked to be printed at last

// hence made to sleep for 3 seconds.

sleep(3);

printf("parent --> pid = %d\n", getpid());

}

}

}

return 0;

}

Output:

child[3]-->pid=50 and ppid=47

child[2]-->pid=49 and ppid=47

child[1]-->pid=48 and ppid=47

parent-->pid=47