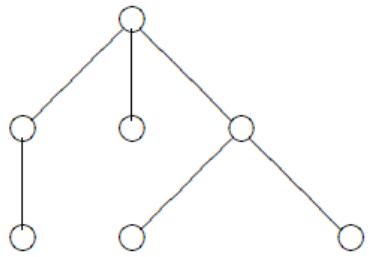
Lebanese University	Operating System II	Partial 2017-2018
Faculty of Sciences I	I3303/INFO324- Solution	Duration: 1 hour

Problem 1: (20 minutes)

A. Write in one statement the code to create the following tree of processes



P.S: **if** and **else** are considered as one statement (nested if else is not permitted)

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>

void main() {
printf("I'm the main with pid=%d\n",getpid());
if (fork())
 fork() && (fork() || fork());
else
 fork() && fork();
while(1);

}
```

B. We consider the following piece of code:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>

void main() {
printf("I'm the main with pid=%d\n",getpid()
if (fork())
 fork() && (fork() || fork());
else
 fork()*fork();
while(1);

}
```

1. How many processes are created after the execution of the above program?
2. Draw the tree of processes generated.

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>

void main() {
printf("I'm the main with pid=%d\n",getpid());
if (fork())
 fork() && (fork() || fork());
else
 fork()*fork();
while(1);

}

/*********************************************************
Ex1-B(2445)─┬─Ex1-B(2446)─┬─Ex1-B(2450)──Ex1-B(2452)
            │             └─Ex1-B(2451)
            ├─Ex1-B(2447)
            └─Ex1-B(2448)──Ex1-B(2449)
*********************************************************/
```

Lebanese University	Operating System II	Partial 2017-2018
Faculty of Sciences I	I3303/INFO324- Solution	Duration: 1 hour

Problem 2: (parts A and B are independent) **(40 minutes)**

A. A parent process creates N parallel child processes that have a point of appointment (RDV). The two processes should execute two functions: **Pre_RDV()** and **RDV()** consecutively. A process arriving at the RDV point (i.e., before executing the function **RDV()**) should waits if there is at least one other process that did not arrived. The last arriving process wakes up all the blocked processes. Write a solution that solves this problem.

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/wait.h>
#include<signal.h>

void handler(int);

void main() {
int i=0,j,pid,p1[2], p2[2]; int N=5, cnt=0;
pipe(p1); pipe(p2); signal(SIGUSR1, handler);
write(p2[1],&cnt,sizeof(int));

while (i<N) {
pid=fork();
if(pid) {write(p1[1],&pid,sizeof(int));}
else {
   if(i<N-1) pause(); // pre_RDV()
   else {
     for(j=0;j<N-1;j++){
          read(p1[0],&pid,sizeof(int));
          kill(pid,SIGUSR1); }
   }
}
i++;
}
} // end main

void handler(int sig) {
printf("child process pid=%d, i received a signal %d\n",getpid(),sig);
// rdv();
exit(1);
}
```

In this part, we suppose that we have a shared data segment that contains a shared variable between **N <u>separated processes</u>** (each process is an independent C program). These processes should access the shared variable and increment it by 1. The shared variable is created by the first process.

A. Write the code of the first process that creates the shared data segment and the shared variable and initialize it to 1. This process should maintain the **<u>way of communicating</u>** the (ID) of the shared data segment between the different processes.

```c
void main() {
int id, fd, key=5;
char *add;
// create named pipe
fd=open("MyPipe",O_CREAT | 0644);

if(fd==-1) printf("error opening pipe\n");
printf("fd=%d\n",fd);

if ((id=shmget(key, sizeof(int),
  IPC_CREAT | IPC_EXCL | 0600))==-1)
 exit(1);

// write id of shared memory into named pipe
write(fd,&id,sizeof(int));

if((add=(char *) shmat(id,NULL,0))==(char *)-1)
  exit(2);

int* val=(int *)add; //shared variable
*val=1; // set the value to 1
}
```

B. Write the code of the other processes (all have similar behavior) that should access the shared data segment via its ID and increment the shared variable and print out on the screen the value of the shared variable along with its pid.

```c
void main() {
int id, fd;
char *add;
fd=open("MyPipe",O_WRONLY | 0644);
if(fd==-1) printf("error opening pipe\n");
read(fd,&id,sizeof(int));
if((add=(char *) shmat(id,NULL,0))==(char *)-1)
  exit(2);

int* val=(int *)add; //shared variable
*val++; // increment the value by 1
}
```