3.2 Including the initial parent process, how many processes are created by the program shown in Program ?

```
#include <stdio.h>
#include <unistd.h>
int main()
{ /* fork a child process */
fork();
/* fork another child process */
fork();
/* and fork another */
fork();
return 0;
}
```

A. The number of processes created including the initial parent is **8**.

3.9 Describe the actions taken by a kernel to context-switch between processes.

A. Switching the CPU from one process to another process requires performing a state save of the current process and a state restore of a different process. This is termed as a context switch. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.

3.12 Including the initial parent process, how many processes are created by the program shown in program ?

```
#include <stdio.h>
#include <unistd.h>
int main()
{ int i;
for (i = 0; i < 4; i++)
fork();
return 0;
}
```

A. 16 processes are created in total including parent process.

3.13 Explain the circumstances under which which, the line of code marked printf("LINE J") in program will be reached.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main()
{pid_t pid;
/* fork a child process */
pid = fork();
if (pid < 0) { /* error occurred */
```

```
fprintf(stderr, "Fork Failed");
return 1;
}else if (pid == 0) { /* child process */
execlp("/bin/ls","ls",NULL);
printf("LINE J");
}else { /* parent process */
/* parent will wait for the child to complete */
wait(NULL);
printf("Child Complete");
}
return 0; }
```

A. The line would be printed only if an error occurs during the calling of execlp**.**

3.17 Using the program shown below, explain what the output will be at lines X and Y.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#define SIZE 5
int nums[SIZE] = {0,1,2,3,4};
int main()
{int i;
pid_t pid;
pid = fork();
if (pid == 0) { for (i = 0; i < SIZE; i++) { nums[i] *= -i;
printf("CHILD: %d ",nums[i]); /* LINE X */
}
}else if (pid > 0) { wait(NULL);
for (i = 0; i < SIZE; i++)
printf("PARENT: %d ",nums[i]); /* LINE Y */
}
return 0;
}
```

A. If the process are child then the output at line X is **CHILD: 0 CHILD: -1 CHILD: -4 CHILD: -9 CHILD: -16**
If the process are parent then the output at line Y is **PARENT: 0 PARENT: 1 PARENT: 2 PARENT: 3 PARENT: 4**

Quiz: How many processes are created by the above program?

```
pid = fork (); //fork #1
pid = fork (); //fork #2
pid = fork (); //fork #3
if (pid == 0)
    {
      fork ();  //fork #4
    }
```

```
fork ();         //fork #5
```

A. 24 process are created because

For the first fork 2 process are created. For the second fork 4 are created and for the third 8 are created. In the next call, new process are created only for the child process and then total of 12 are created. Finally 24 are created from the 12.