1.Fork is?

1. the dispatching of a task
2. the creation of a new job
3. increasing the priority of a task
4. the creation of a new process

Sol: 4

Explanation
In computing, particularly in the context of the Unix operating system and its workalikes, fork is an operation whereby a process creates a copy of itself. It is usually a system call, implemented in the kernel. Fork is the primary method of process creation on Unix-like operating systems.

2.Output?

```
main()
{
   if(fork()>0)
   sleep(100);
}
```

1. an orphan process
2. a zombie process
3. a process that executes forever
4. none of the mentioned

Sol: c

3.Which of the following system calls does not return control to the calling point on termination?

1. fork
2. exec
3. ioctl
4. Longjmp

Sol : exec

4.What is the output of the following code?

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

int main()
{
    if (fork() && (!fork())) {
        if (fork() || fork()) {
            fork();
        }
    }
    printf("2 ");
    return 0;
}
```

Sol: 2 2 2 2 2 2 2
Explanation
1. Fork will create two processes: one parent P (has process id of new child) and other one is child C1 (process id=0).

2. In the if statement we are using the AND operator (i.e, &&) and in this case if the first condition is false then it will not evaluate the second condition and print 2. Parent process P checks for the second condition and creates two new processes (one parent P and other is child C2). In the second condition we are using the NOT operator which returns true for child process C2 and it executes the inner if statement.

3. Child C2 again creates two new processes (one parent C2 and child C3) and we are using OR operator (i.e, ||) which evaluates the second condition when the first condition is false. Parent C2 executes if part and creates two new processes (one parent C2 and child C4) whereas child C3 checks for second condition and creates two new processes (one parent C3 and child C5).

4. Parent C3 enters if part and further creates two new processes (one parent C3 and child C6).

5.What is the output of the following code?

```c
#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;
}
```

Ans 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 checks next 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).

6.What is the output of the following code?

```c
#include <stdio.h>
#include <unistd.h>
int main()
{
    if (fork() || fork())
        fork();
    printf("1 ");
    return 0;
}
```

Sol : 1 1 1 1 1

Explanation

1. It will create two processes: one parent P (has the process ID of the child process)and the other is child C1 (process ID = 0).
2. In the if statement we used the OR operator(||) and in this case the second condition is evaluated when the first condition is false.
3. Parent process P will return a 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 the 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