National University of Computer and Emerging Sciences, Lahore Campus

Course:	Operating System	Course Code:	CS-2006
Quiz:	Quiz:1 sample 1	Marks	10

Name:

Q:1- Which of the following actions by a running process will always result in a context switch of the running process, even in a non-preemptive kernel design? [2]

(a) Servicing a disk interrupt, that results in another blocked process being marked as ready/runnable.

(b) A blocking system call.

(c) The system call exit, to terminate the current process.

(d) Servicing a timer interrupt.

Q:2- Consider the following program which uses the fork()and wait() system calls. What is the result written by the program to the standard output stream? Lets suppose parent process have process id= 123 and child process have process id= 345.

[3]

```
int main()
{
        int i;
        pid_t pid;
        pid = fork();
        if(pid > 0)
        {
        pid_t w1=wait(NULL);
        printf ("value is %d \n" , w1);
        printf ("Parent starts\n ");
        for (i=10; i>=2; i-=2)
                printf ("%d\n",i);
        printf ("\nParent ends\n");
        }

        else if (pid == 0)
        {
        printf ("Child starts\n ");
        for (i=10; i>=1; i-=2)
                printf ("%d \n",i);
        printf ("\nChild ends\n");
        exit(0);
        }
}
```

Output:

Child starts
10
8
6
4
2
Child ends
Value is 345
Parent starts
10
8
6
4
2
Parent ends

2025.02.21 22:57

CamScanner

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Operating System	Course Code:	CS-2006
Quiz:	Quiz:1 sample 2	Marks	10
Name:		Roll No.	

Q:1- Consider a process P that needs to save its CPU execution context (values of some CPU registers) on some stack when it makes a function call or system call. Which of the following statements is/are true? [2]

(a) During a system call, when transitioning from user mode to kernel mode, the context of the process is saved on its kernel stack.
(b) During a function call in user mode, the context of the process is saved on its user stack.
(c) During a function call in kernel mode, the context of the process is saved on its user stack.
(d) During a function call in kernel mode, the context of the process is saved on its kernel stack.

Q:2- Consider the following program which uses the fork()and wait() system calls. What is the result written by the program to the standard output stream? Lets suppose parent process have process id= 123 and child process have process id= 345. [3]

```
int main()
{
        int i;
        pid_t pid;
        pid = fork();
        if(pid > 0)
        {
        pid_t w1=wait(NULL);
        printf ("value is %d \n" , w1);
        printf ("Parent starts\n ");
        for (i=10; i<=2; i-=2)
                printf ("%d\n",i);
        printf ("\nParent ends\n");
        }

        else if (pid == 0)
        {
        printf ("Child starts\n ");
        for (i=10; i<11; i-=2)
                printf ("%d \n",i);
        printf ("\nChild ends\n");
        exit(0);
        }
}
```

Output:

child starts
10
8
6
4
2
child ends

value is 345
Parent starts
10
8
6
4
2
Parent ends

CamScanner

Q3:- Consider the following program which uses the fork() system call. What is the result written by the program to the standard output stream? What is the resulting process tree? **[2+3]**

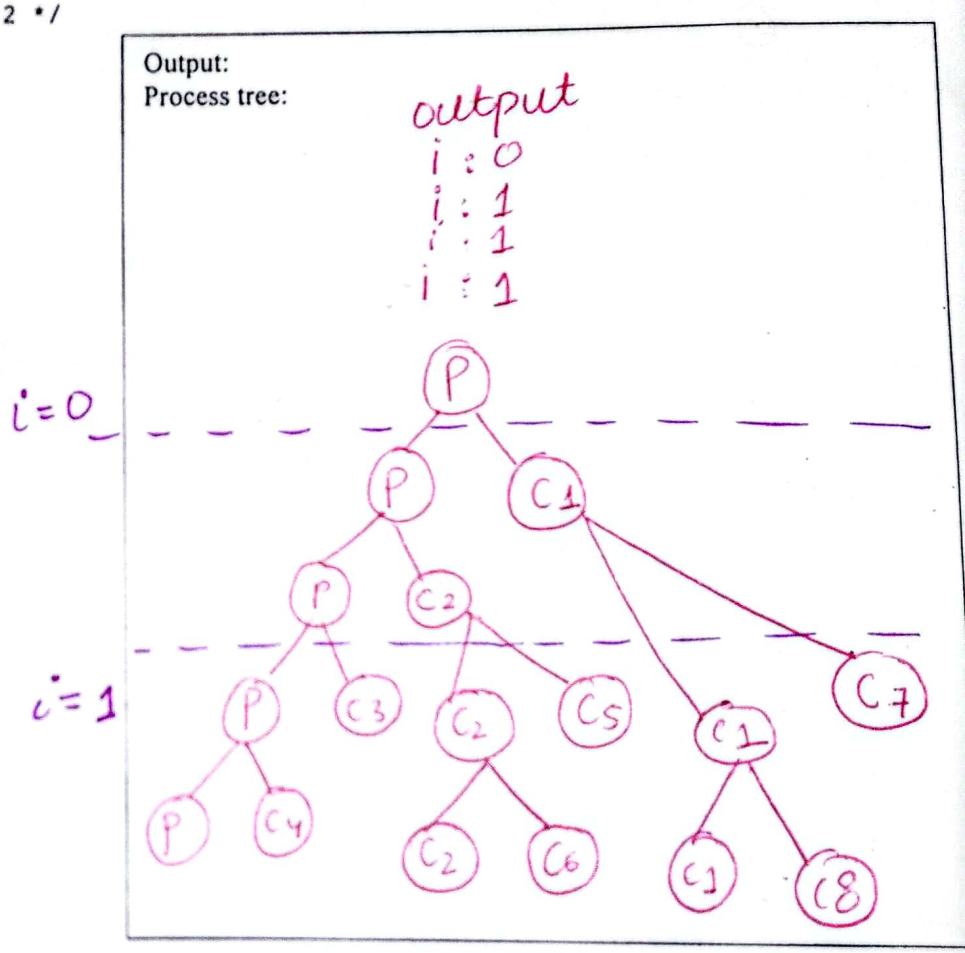
```
#include <stdio.h>

int i:

int main ()
{
for (i=0 ; i < 2 ; i++)
     {
     printf("i: %d \n", i);
     if (fork()) /* call #1 */
          fork(); /* call #2 */
     }
}
```

Output:

Process tree:



2025 02 21 22:58

CamScanner