Name: Girish S Roll No.: AM.EN.U4AIE22044

OPERATING SYSTEMS

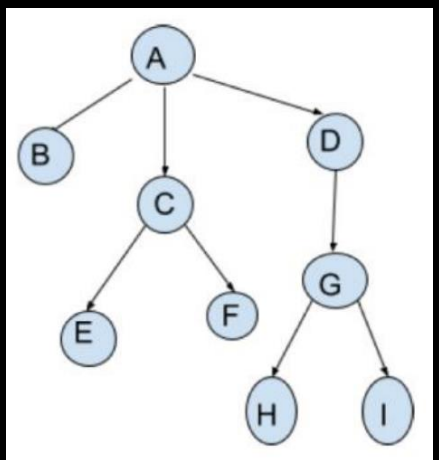
1. Execute the above program more than once. What is the order in which the processes are being executed? Is it the same in every execution?

1.

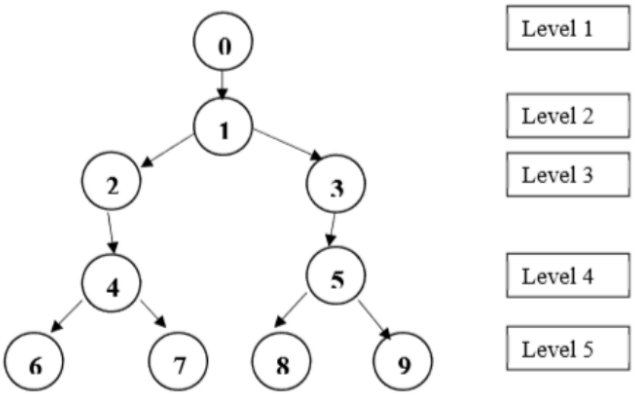
```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

int main(){
    pid_t pid = getpid();
    pid_t ppid = getppid();
    printf("Label -> A PID -> %d PPID -> %d\n", getpid(), getppid());
    if(fork()){
        wait(NULL);
        if(fork()){
            wait(NULL);
            if(!fork()){
                printf("label -> D PID ->  %d PPID -> %d\n", getpid(), getppid());
                if(!fork()){
                    printf("label -> G PID -> %d PPID -> %d\n", getpid(), getppid());
                    if(fork()){
                        wait(NULL);
                        if(!fork()){printf("label -> I PID -> %d PPID -> %d\n", getpid(), getppid());}
                        else{wait(NULL);}}
                    else { printf("label -> H PID -> %d PPID -> %d\n", getpid(), getppid()); }}
                else { wait(NULL); }}
            else { wait(NULL); }}
        else{
            printf("label -> C PID -> %d PPID -> %d\n", getpid(), getppid());
            if(fork()){
                wait(NULL);
                if(!fork()) { printf("label -> F PID -> %d PPID -> %d\n", getpid(), getppid()); }
                else { wait(NULL); }}
            else { printf("label -> E PID -> %d PPID -> %d\n", getpid(), getppid()); }}}
    else { printf("label -> B PID -> %d PPID -> %d\n", getpid(), getppid()); }
    return 0; }
```

```
root@Giriirig:~# ./q1
Label -> A PID -> 3669 PPID -> 364
label -> B PID -> 3670 PPID -> 3669
label -> C PID -> 3671 PPID -> 3669
label -> E PID -> 3672 PPID -> 3671
label -> F PID -> 3673 PPID -> 3671
label -> D PID ->  3674 PPID -> 3669
label -> G PID -> 3675 PPID -> 3674
label -> H PID -> 3676 PPID -> 3675
label -> I PID -> 3677 PPID -> 3675
root@Giriirig:~# gcc q1.c -o q1
root@Giriirig:~# ./q1
Label -> A PID -> 3695 PPID -> 364
label -> B PID -> 3696 PPID -> 3695
label -> C PID -> 3697 PPID -> 3695
label -> E PID -> 3698 PPID -> 3697
label -> F PID -> 3699 PPID -> 3697
label -> D PID ->  3700 PPID -> 3695
label -> G PID -> 3701 PPID -> 3700
label -> H PID -> 3702 PPID -> 3701
label -> I PID -> 3703 PPID -> 3701
root@Giriirig:~#
```

Yes, the order is same in every execution
A->B->C->E->F->D->G->G->H->I
And it follows the process tree, only the Process ID gets reassigned every time



2. Write a program to create processes according to the tree structure given below. All processes should print their Process id and Parent Process id and the label given in the diagram.



2.

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main(){
    printf("Label -> 0 PID -> %d PPID -> %d\n", getpid(), getppid());
    if(!fork()){ // process 1
        printf("Label -> 1 PID -> %d PPID -> %d\n", getpid(), getppid());
        if(!fork()){ // process 2
            printf("Label -> 2 PID -> %d PPID -> %d\n", getpid(), getppid());
            if(!fork()){ // process 4
                printf("Label -> 4 PID -> %d PPID -> %d\n", getpid(), getppid());
                if(!fork()){ // process 6
                    printf("Label -> 6 PID -> %d PPID -> %d\n", getpid(), getppid());}
                else{ // process 4
                    wait(NULL);
                    if(!fork()){ // process 7
                        printf("Label -> 7 PID -> %d PPID -> %d\n", getpid(), getppid());}
                    else{wait(NULL);}}}
            else{wait(NULL);}} // process 2
        else{ // process 1
            wait(NULL);
            if(!fork()){ // process 3
                printf("Label -> 3 PID -> %d PPID -> %d\n", getpid(), getppid());
                if(!fork()){ // process 5
                    printf("Label -> 5 PID -> %d PPID -> %d\n", getpid(), getppid());

                    if(!fork()){ // process 8
                        printf("Label -> 8 PID -> %d PPID -> %d\n", getpid(), getppid());}
                    else{ // process 5
                        wait(NULL);
                        if(!fork()){printf("Label -> 9 PID -> %d PPID -> %d\n", getpid(), getppid());} // process 9
                        else {wait(NULL);}}} // process 8
                else{wait(NULL);}} // process 3
            else{wait(NULL);}}}  // process 1
    else {wait(NULL);} // process 0
    return 0;}
```

```
root@Giriirig:~# gcc lab4q2.c -o q2
root@Giriirig:~# ./q2
Label -> 0 PID -> 4001 PPID -> 364
Label -> 1 PID -> 4002 PPID -> 4001
Label -> 2 PID -> 4003 PPID -> 4002
Label -> 4 PID -> 4004 PPID -> 4003
Label -> 6 PID -> 4005 PPID -> 4004
Label -> 7 PID -> 4006 PPID -> 4004
Label -> 3 PID -> 4007 PPID -> 4002
Label -> 5 PID -> 4008 PPID -> 4007
Label -> 8 PID -> 4009 PPID -> 4008
Label -> 9 PID -> 4010 PPID -> 4008
root@Giriirig:~#
```

3. Write a program to find the area and perimeter of circle and square. Create separate processes to perform the calculation of circle and square.

3.

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int r;
    float Carea, Cperimeter;

    printf("Enter the radius of the circle: ");
    scanf("%d" , &r);

    int a;
    printf("Enter the side of the square: ");
    scanf("%d" ,&a);

    if(fork()){
        printf("Label -> CIRCLE PID -> %d PPID -> %d\n" , getpid() , getppid());
        Carea = 3.14 * r * r;
        Cperimeter = 2 * 3.14 * r;
        printf("The area of circle is %f and perimeter is %f\n", Carea, Cperimeter);
        }
    else{
        printf("Label -> SQUARE PID -> %d PPID -> %d\n", getpid(), getppid());
        printf("Area of square is %d and perimeter is %d\n", (a*a), (4 * a));
    }
    return 0;}
```

```
root@Giriirig:~# gcc lab4q3.c -o q3
root@Giriirig:~# ./q3
Enter the radius of the circle: 10
Enter the side of the square: 6
Label -> CIRCLE PID -> 4016 PPID -> 364
Label -> SQUARE PID -> 4017 PPID -> 4016
Area of square is 36 and perimeter is 24
The area of circle is 314.000000 and perimeter is 62.799999
root@Giriirig:~#
```

4. Modify the above program as follows: The parent process should create two children.
[User enters Value of variable 'a' only once]. The first child finds the area and perimeter of a circle with radius 'a'. The Second child finds the area and perimeter of square with side 'a'.

4.

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

int main(){
    int a;
    printf("Enter the Value for a: ");
    scanf("%d" ,&a);

    if(!fork()){
        printf("Label -> CIRCLE PID -> %d PPID -> %d\n" , getpid() , getppid());
        printf("The area of circle is %f and perimeter is %f\n", 3.14*a*a, 2*3.14*a);
        }
    else{
        if(!fork()){
            printf("Label -> SQUARE PID -> %d PPID -> %d\n", getpid(), getppid());
            printf("Area of square is %d and perimeter is %d\n", (a*a), (4 * a));
        }
        else{wait(NULL);} // parent has to wait until both children finishes,
    }                     // otherwise child will force exits with incomplete output
    return 0;}
```

```
root@Giriirig:~# gcc lab4q4.c -o q4
root@Giriirig:~# ./q4
Enter the Value for a: 10
Label -> CIRCLE PID -> 4024 PPID -> 4023
The area of circle is 314.000000 and perimeter is 62.800000
Label -> SQUARE PID -> 4025 PPID -> 4023
Area of square is 100 and perimeter is 40
root@Giriirig:~#
```

5. Modify the previous program to make the parent process wait until the completion of its children. **[Hint. Use wait() system call]**

5.

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

int main(){
    int a;
    printf("Enter the Value for a: ");
    scanf("%d" ,&a);

    if(!fork()){ // circle child
        printf("Label -> CIRCLE PID -> %d PPID -> %d\n" , getpid() , getppid());
        printf("The area of circle is %f and perimeter is %f\n", 3.14*a*a, 2*3.14*a);
        }
    else{
        wait(NULL);
        if(!fork()){ // square child
            printf("Label -> SQUARE PID -> %d PPID -> %d\n", getpid(), getppid());
            printf("Area of square is %d and perimeter is %d\n", (a*a), (4 * a));
        }
        else{wait(NULL);}
    } // order is always circle -> square
    return 0;}
```

```
root@Giriirig:~# gcc lab4q5.c -o q5
root@Giriirig:~# ./q5
Enter the Value for a: 10
Label -> CIRCLE PID -> 4043 PPID -> 4042
The area of circle is 314.000000 and perimeter is 62.800000
Label -> SQUARE PID -> 4044 PPID -> 4042
Area of square is 100 and perimeter is 40
root@Giriirig:~#
```

6. Create a parent process having two children. The first child should overwrite its address space with a process that prints "Happy new year" (happynewyear.c).
The second child should overwrite its address space with another process that prints the sum of digits of a number entered by the user(sum.c). **[Hint: use exec family of system calls]**

Sample output: The output should come in the following order

Happy new year

Enter the number: 123
Sum of Digits: 6
Parent exiting ...good bye.

6.

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

int main(){
    if(!fork()){ // first child
        execl("./happynewyear","./happynewyear", NULL);
    }
    else{
        wait(NULL);
        if(!fork()){ // second child waits until first child finishes
            execl("./sum","./sum",NULL);
        }
        else{ // parent waits until both children finishes
            wait(NULL);
            printf("Parent process exiting");
        }
    }
    return 0;
}
```

```
root@Giriirig:~# gcc -o happynewyear happynewyear.c
root@Giriirig:~# gcc -o sum sum.c
root@Giriirig:~# gcc lab4q6.c -o q6
root@Giriirig:~# ./q6
Happy New Year
Please enter a number: 1234
Sum of the digits: 10
Parent process exiting
root@Giriirig:~#
```