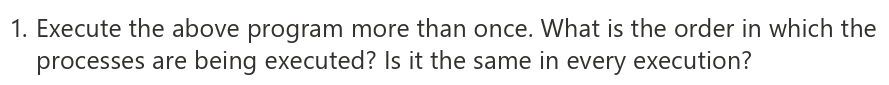
**22AIE202 – OPERATING SYSTEMS**

**LABSHEET 4**

Name : Anuvind MP

Roll no: AM.EN.U4AIE22010

--------------------------------------------------------------------------------------------------------------------------



**CODE :**

#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()); }

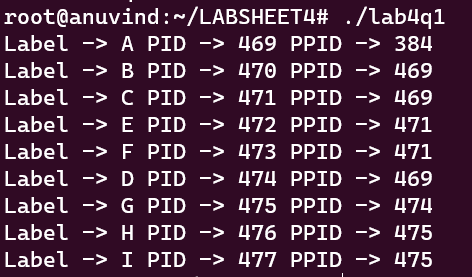
                else { wait(NULL); }}

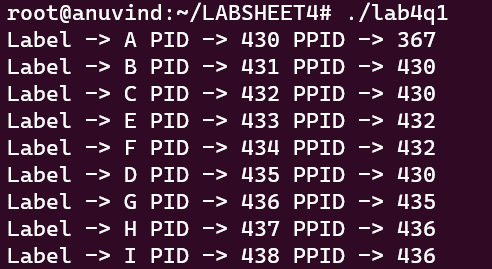
            else { printf("label -> E PID -> %d PPID -> %d\n", getpid(), getppid()); }}}

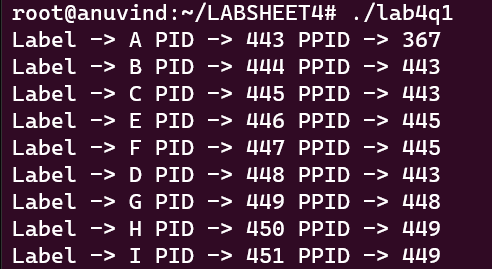
    else{printf("label -> B PID -> %d PPID -> %d\n", getpid(), getppid());}

    return 0;}

**OUTPUT :**

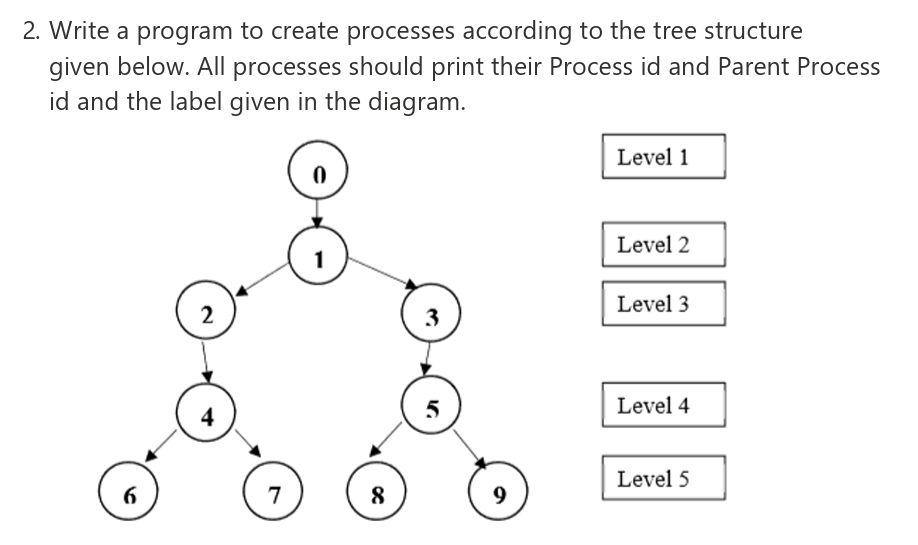
****

****

****

**INFERENCE :**

The sequence of process execution remains consistent in the provided code, yet each time it runs, the process IDs vary. This means that while the order of operations remains unchanged, the specific identification numbers assigned to each process differ with each execution.

****

**CODE :**

#include <stdio.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

    pid\_t pid = getpid();

    pid\_t ppid = getppid();

    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

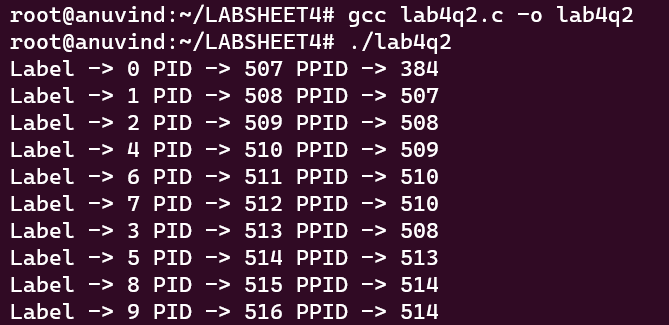
    }

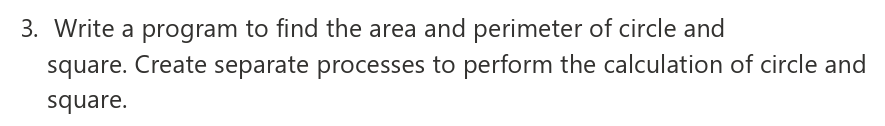
    else {wait(NULL);} // process 0

    return 0;

}

**OUTPUT :**

****

****

**CODE:**

#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{

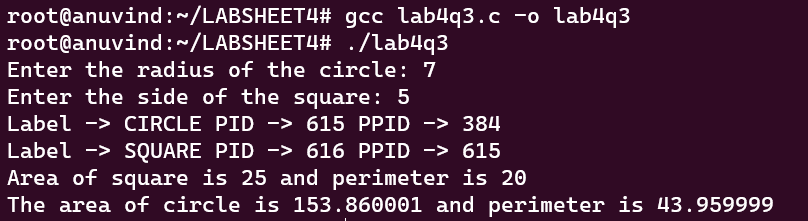
        printf("Label -> SQUARE PID -> %d PPID -> %d\n", getpid(), getppid());

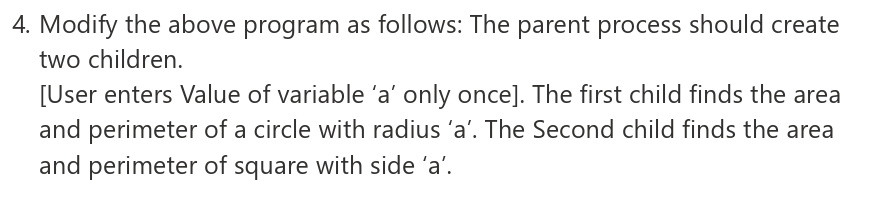
        printf("Area of square is %d and perimeter is %d\n", (a\*a), (4 \* a));

    }

    return 0;}

**OUTPUT:**

****

****

**CODE :**

#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));

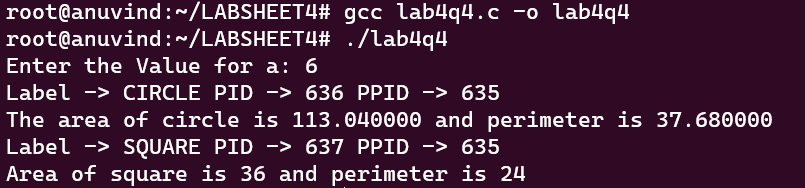
        }

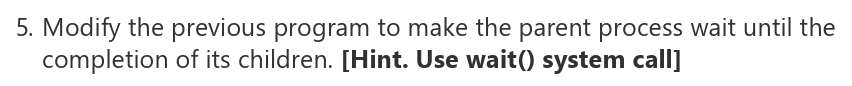
        else{wait(NULL);} // parent has to wait until both children finishes, otherwise child will force exits with incomplete output

    }

    return 0;}

**OUTPUT :**

****

****

**CODE :**

#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));

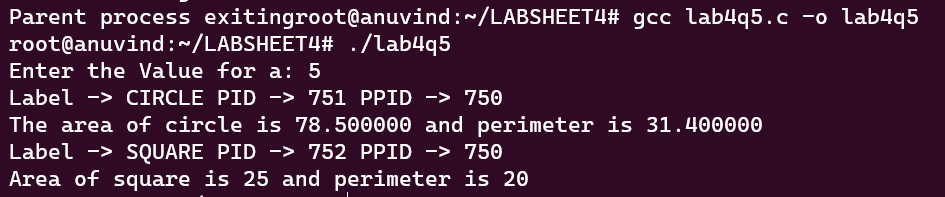
        }

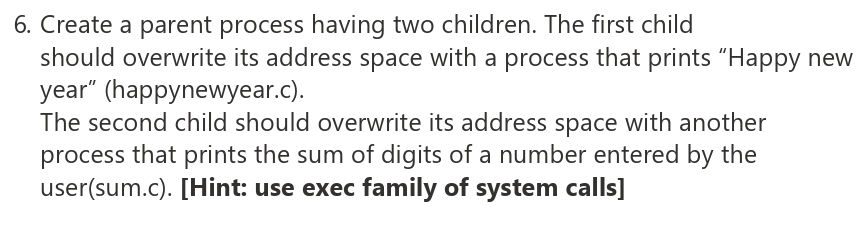
        else{wait(NULL);}

    } // order is always circle -> square

    return 0;}

**OUTPUT :**

****

****

**CODE :**

#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 ....Good bye..");

        }

    }

    return 0;

}

**CODE (sum.c) :**

#include <stdio.h>

#include <unistd.h>

int main(){

   printf("Please enter a number: ");

   int num;

   scanf("%d", &num);

   int sum=0;

   while(num>0){

        int k=num%10;

        sum+=k;

        num=num/10;

   }

    printf("Sum of the digits: %d\n", sum);

    return 0;

}

**CODE (Happynewyear):**

#include <stdio.h>

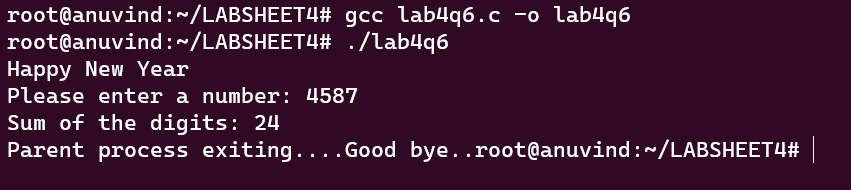
int main(){

    printf("Happy New Year\n");

    return 0;

}

**OUTPUT :**

****