Lab 5: Process creation

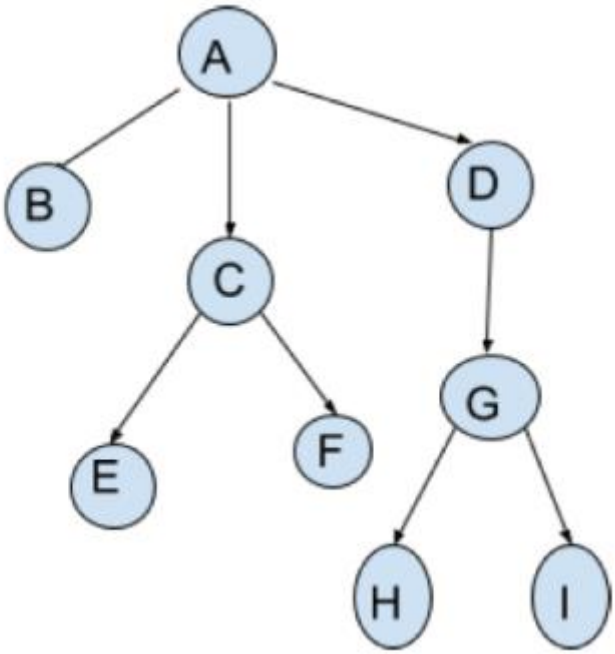
The following C program demonstrates the use of getpid () and getppid () to print the PID of the process and the PID of its parent process respectively.

Header files to be included:

1. stdio.h - it is used for printf() function
2. sys/types.h - it is used for pid_t type, that is the data type of the variables which are
using to store the process ids.
3. unistd.h - it is used for getpid() and getppid() functions and also for fork() system call.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
   //variable to store calling function's process id
     pid_t process_id;
   //variable to store parent function's process id
     pid_t p_process_id;
   //getpid() - will return process id of calling function
     process_id = getpid();
  //getppid() - will return process id of parent function
     p_process_id = getppid();
  //printing the process ids
     printf ("The process id: %d\n",process_id);
     printf("The process id of parent function: %d\n”, p_process_id);
     return0;
}
```

A program to create processes according to the tree structure given below.



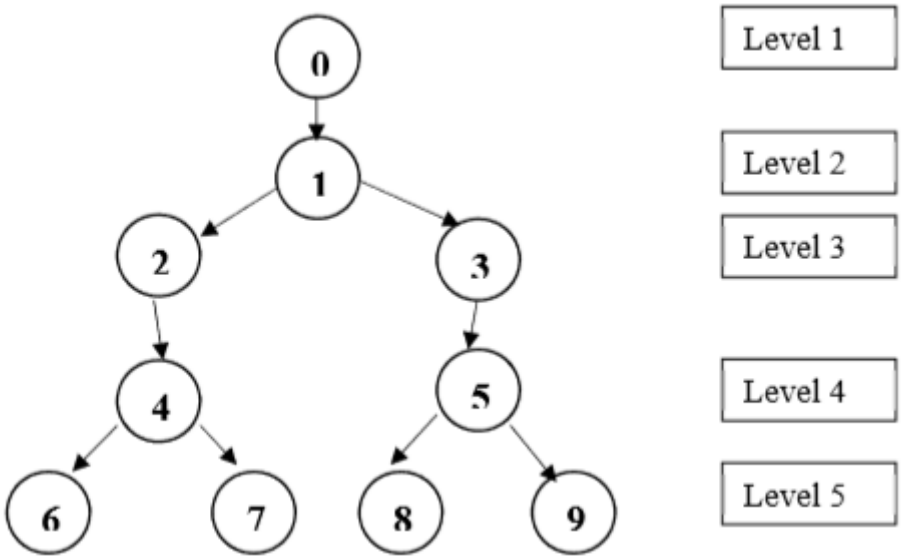

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

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;
}
```

```
examuser@15CPU0120L:~/Desktop/saran$ gcc -o 1.o 1.c
1.c: In function 'main':
1.c:10:3: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
   wait(NULL);
   ^
examuser@15CPU0120L:~/Desktop/saran$ ./1.o
Label -> A PID -> 6694 PPID -> 4960
label -> B PID -> 6695 PPID -> 6694
label -> C PID -> 6696 PPID -> 6694
label -> E PID -> 6697 PPID -> 6696
label -> F PID -> 6698 PPID -> 6696
Label -> D PID -> 6699 PPID -> 6694
label -> G PID -> 6700 PPID -> 6699
label -> H PID -> 6701 PPID -> 6700
label -> I PID -> 6702 PPID -> 6700
```

Questions

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?
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.



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.
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'.

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

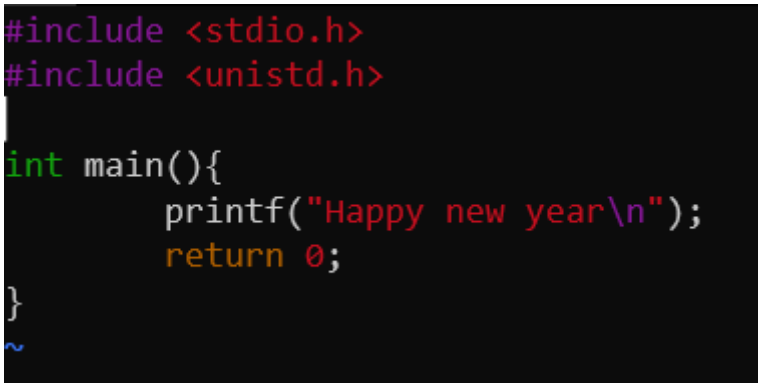
Happy new year

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

```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 Digits: %d\n",sum);
        return 0;
}
~
```

sum.c



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

int main(){
        printf("Happy new year\n");
        return 0;
}
~
```

happynewyear.c