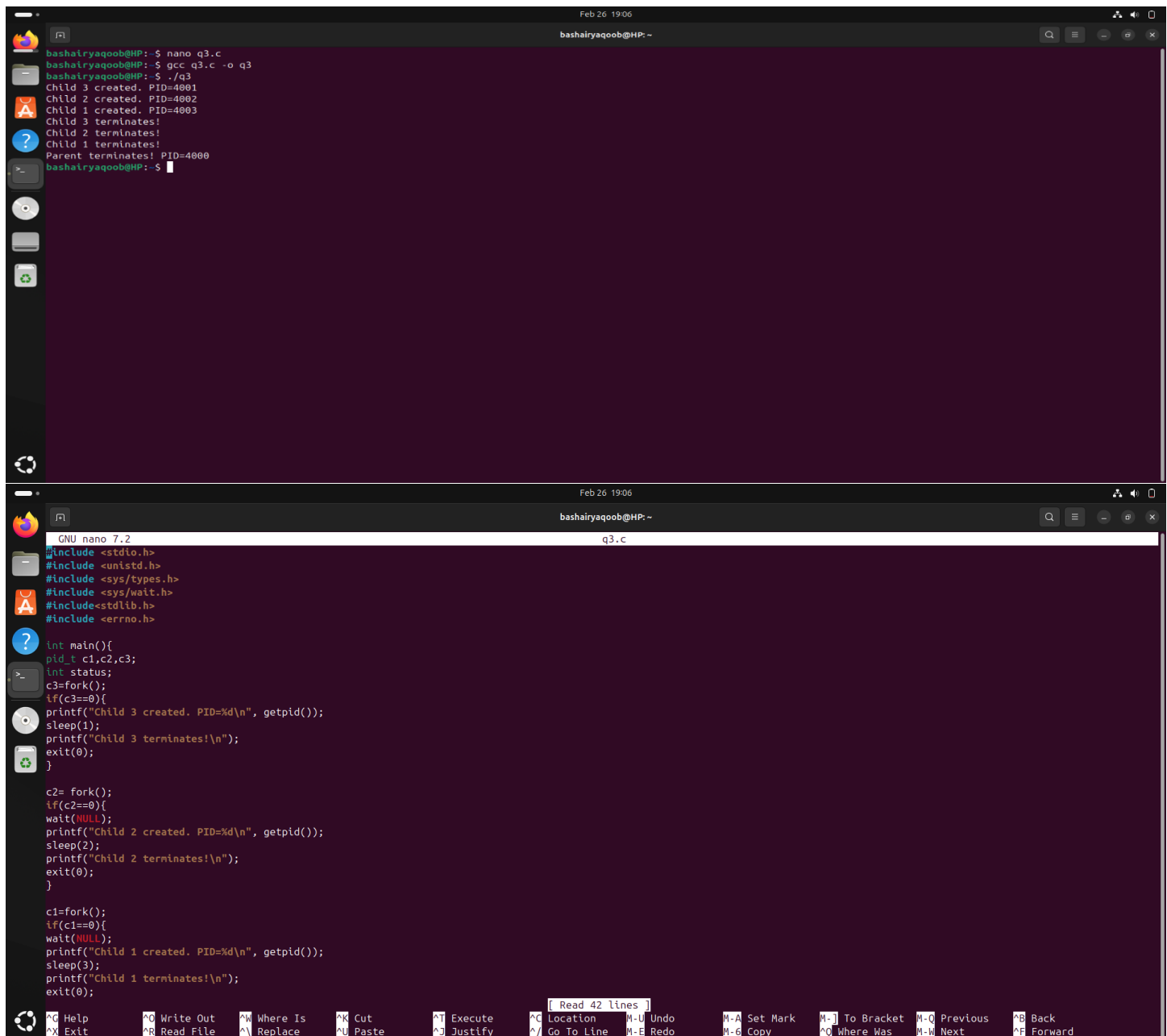


## OS LAB 4:

24k-0810

Q3: 3. Write Program to create four processes (1 parent and 3 children) where they terminate in a sequence as follows:

- Parent process terminates at last
- The first child terminates before the parent and after the second child.
- The second child terminates after the last and before the first child.
- The third child terminates first.



The image consists of two screenshots of a terminal window. The top screenshot shows the execution of a C program. The output is as follows:

```
bashairyaqoob@HP:~$ nano q3.c
bashairyaqoob@HP:~$ gcc q3.c -o q3
bashairyaqoob@HP:~$ ./q3
Child 3 created. PID=4001
Child 2 created. PID=4002
Child 1 created. PID=4003
Child 3 terminates!
Child 2 terminates!
Child 1 terminates!
Parent terminates! PID=4000
bashairyaqoob@HP:~$
```

The bottom screenshot shows the source code of the program in the nano editor:

```
GNU nano 7.2 q3.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <errno.h>

int main(){
    pid_t c1,c2,c3;
    int status;
    c3=fork();
    if(c3==0){
        printf("Child 3 created. PID=%d\n", getpid());
        sleep(1);
        printf("Child 3 terminates!\n");
        exit(0);
    }

    c2= fork();
    if(c2==0){
        wait(NULL);
        printf("Child 2 created. PID=%d\n", getpid());
        sleep(2);
        printf("Child 2 terminates!\n");
        exit(0);
    }

    c1=fork();
    if(c1==0){
        wait(NULL);
        printf("Child 1 created. PID=%d\n", getpid());
        sleep(3);
        printf("Child 1 terminates!\n");
        exit(0);
    }
}
```

The terminal window has a status bar at the bottom with various shortcuts: Help, Write Out, Where Is, Cut, Execute, Location, M-U Undo, M-A Set Mark, M-] To Bracket, M-Q Previous, ^B Back, Exit, Read File, Replace, Paste, Justify, Go To Line, M-E Redo, M-G Copy, M-^ Where Was, M-W Next, and ^F Forward.

```
GNU nano 7.2 q3.c
int status;
c3=fork();
if(c3==0){
printf("Child 3 created. PID=%d\n", getpid());
sleep(1);
printf("Child 3 terminates!\n");
exit(0);
}

c2= fork();
if(c2==0){
wait(NULL);
printf("Child 2 created. PID=%d\n", getpid());
sleep(2);
printf("Child 2 terminates!\n");
exit(0);
}

c1=fork();
if(c1==0){
wait(NULL);
printf("Child 1 created. PID=%d\n", getpid());
sleep(3);
printf("Child 1 terminates!\n");
exit(0);
}

wait(&status);
wait(&status);
wait(&status);
printf("Parent terminates! PID=%d\n", getpid());
return 0;
}
```

Q4: 4. Write a program which creates 4 processes for parallel programming. Each parent will wait for the termination of their child.

```
bashairyaqoob@HP:~$ nano q4.c
bashairyaqoob@HP:~$ gcc q4.c -o q4
bashairyaqoob@HP:~$ ./q4
Child 1: PID=4765 PPID=4763
Child 1 terminates!
Parent PID 4763 terminates
Child 2: PID=4766 PPID=4763
Child 2 terminates!
Parent PID 4763 terminates
Child 3: PID=4767 PPID=4763
Child 3 terminates!
Parent PID 4763 terminates
Child 4: PID=4768 PPID=4763
Child 4 terminates!
Parent PID 4763 terminates
bashairyaqoob@HP:~$
```

```
GNU nano 7.2 q4.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

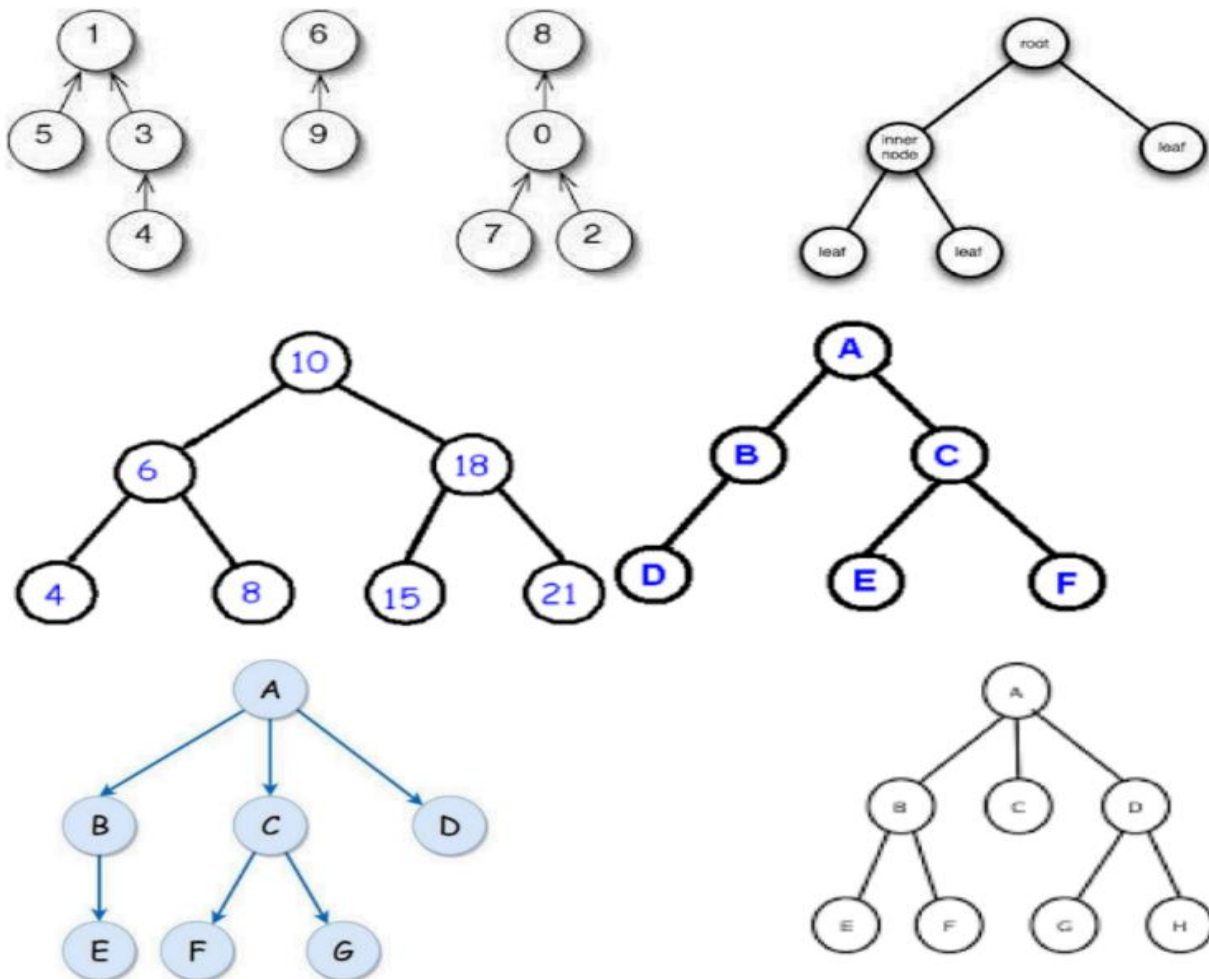
int main(){
    pid_t child;
    int i, status;
    for(i=0; i<4; i++){
        child=fork();

        if(child<0){
            perror("Fork failed!");
            exit(1);
        }

        if(child==0){
            printf("Child %d: PID=%d PPID=%d\n", i+1, getpid(), getppid());
            sleep(2);
            printf("Child %d terminates!\n", i+1);
            exit(0);
        }

        else{
            wait(&status);
            printf("Parent PID %d terminates\n", getpid());
        }
    }
    return 0;
}
```

Q5: Implement the following 6 tree structures. Each node must print its name and PID.



C:

```
bashairyaqoob@HP: ~  
bashairyaqoob@HP:~$ nano q5c.c  
bashairyaqoob@HP:~$ gcc q5c.c -o q5c  
bashairyaqoob@HP:~$ ./q5c  
I am process 8 and my PID is 5124  
I am process 0 and my PID is 5125  
I am process 7 and my PID is 5126  
I am process 2 and my PID is 5127  
bashairyaqoob@HP:~$
```

```
GNU nano 7.2 q5c.c *  
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <sys/wait.h>  
  
int main(){  
    printf("I am process 8 and my PID is %d\n", getpid());  
    pid_t p0=fork();  
    if(p0==0){  
        printf("I am process 0 and my PID is %d\n", getpid());  
        pid_t p7=fork();  
        if(p7==0){  
            printf("I am process 7 and my PID is %d\n", getpid());  
            exit(0);  
        }  
        pid_t p2=fork();  
        if(p2==0){  
            printf("I am process 2 and my PID is %d\n", getpid());  
            exit(0);  
        }  
        wait(NULL);  
        wait(NULL);  
        exit(0);  
    }  
    wait(NULL);  
    return 0;  
}
```

^G Help   ^O Write Out   ^W Where Is   ^K Cut   ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-J To Bracket   M-Q Previous   ^B Back  
^X Exit   ^R Read File   ^\ Replace   ^U Paste   ^J Justify   ^\_ Go To Line   M-E Redo   M-6 Copy   ^Q Where Was   M-W Next   ^F Forward

(D):

The image consists of two terminal screenshots. The top screenshot shows the GNU nano 7.2 editor editing a file named q5d.c. The code is a C program that uses fork() to create a tree of processes. The root process prints its PID, then forks an inner process, which in turn forks three leaf processes. Each process prints its PID before exiting. The bottom screenshot shows the terminal output after compiling and running the program. It displays the execution of nano q5d.c, compilation with gcc, and the execution of ./q5d, which produces the expected output of process PIDs.

```
GNU nano 7.2 q5d.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main(){
    printf("I am Process root PID=%d\n", getpid());
    pid_t inner=fork();
    if(inner==0){
        printf("I am Process inner PID=%d\n", getpid());
        pid_t l1=fork();
        if(l1==0){
            printf("I am Process leaf PID=%d\n", getpid());
            exit(0);
        }
        pid_t l2=fork();
        if(l2==0){
            printf("I am Process leaf PID=%d\n", getpid());
            exit(0);
        }
        wait(NULL);
        wait(NULL);
        exit(0);
    }
    pid_t l3=fork();
    if(l3==0){
        printf("I am Process leaf PID=%d\n", getpid());
        exit(0);
    }
    wait(NULL);
    wait(NULL);
    return 0;
}
```

```
bashairyaqoob@HP: ~
bashairyaqoob@HP:~$ nano q5d.c
bashairyaqoob@HP:~$ gcc q5d.c -o q5d
bashairyaqoob@HP:~$ ./q5d
I am Process root PID=5322
I am Process leaf PID=5324
I am Process inner PID=5323
I am Process leaf PID=5325
I am Process leaf PID=5326
bashairyaqoob@HP:~$
```

(E):

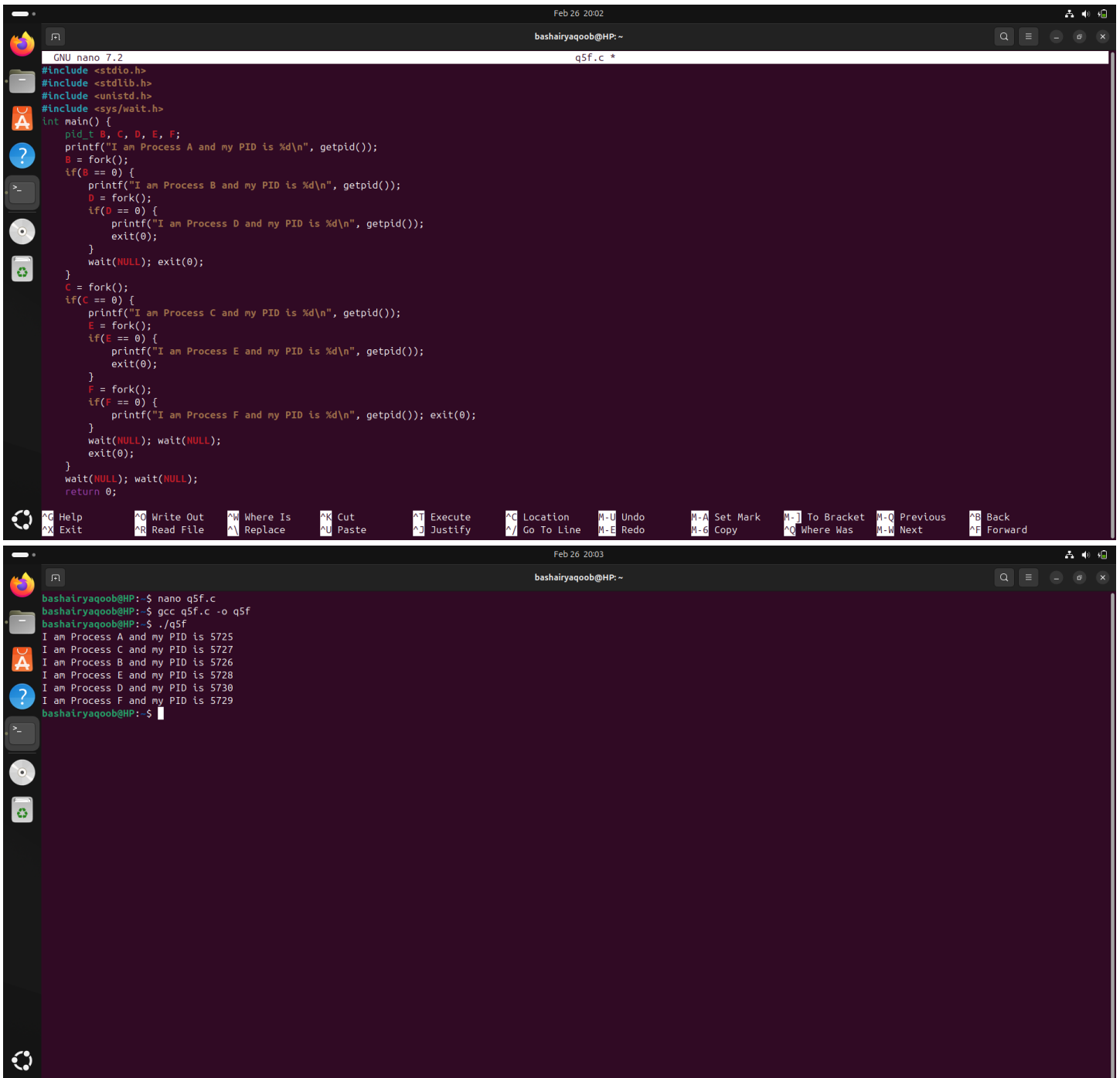
The image consists of two terminal screenshots. The top screenshot shows the GNU nano 7.2 editor editing a file named q5e.c. The code is a C program that uses fork() to create multiple processes. It prints the process ID (PID) for each process. The code is as follows:

```
#include <sys/wait.h>
int main(){
    printf("I am Process 10 PID=%d\n", getpid());
    pid_t p6=fork();
    if(p6==0){
        printf("I am Process 6 PID=%d\n", getpid());
        if(fork()==0){
            printf("I am Process 4 PID=%d\n",getpid());
            exit(0);
        }
        if(fork()==0){
            printf("I am Process 8 PID=%d\n",getpid());
            exit(0);
        }
        wait(NULL); wait(NULL);
        exit(0);
    }
    pid_t p18=fork();
    if(p18==0){
        printf("I am Process 18 PID=%d\n", getpid());
        if(fork()==0){
            printf("I am Process 15 PID=%d\n",getpid());
            exit(0);}
        if(fork()==0){
            printf("I am Process 21 PID=%d\n",getpid());
            exit(0);}
        wait(NULL); wait(NULL);
        exit(0);
    }
    wait(NULL); wait(NULL);
    return 0;
}
```

The bottom screenshot shows the execution of the program. The user runs the command `q5e.c` in the terminal, which compiles the program. Then, the user runs the command `./q5e`, which executes the program. The output shows the process IDs for each process:

```
bashairyaqoob@HP:~$ nano q5e.c
bashairyaqoob@HP:~$ gcc q5e.c -o q5e
bashairyaqoob@HP:~$ ./q5e
I am Process 10 PID=5578
I am Process 6 PID=5579
I am Process 4 PID=5580
I am Process 18 PID=5581
I am Process 8 PID=5582
I am Process 21 PID=5584
I am Process 15 PID=5583
bashairyaqoob@HP:~$
```

(F):



```
GNU nano 7.2 q5f.c *
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t B, C, D, E, F;
    printf("I am Process A and my PID is %d\n", getpid());
    B = fork();
    if(B == 0) {
        printf("I am Process B and my PID is %d\n", getpid());
        D = fork();
        if(D == 0) {
            printf("I am Process D and my PID is %d\n", getpid());
            exit(0);
        }
        wait(NULL); exit(0);
    }
    C = fork();
    if(C == 0) {
        printf("I am Process C and my PID is %d\n", getpid());
        E = fork();
        if(E == 0) {
            printf("I am Process E and my PID is %d\n", getpid());
            exit(0);
        }
        F = fork();
        if(F == 0) {
            printf("I am Process F and my PID is %d\n", getpid()); exit(0);
        }
        wait(NULL); wait(NULL);
        exit(0);
    }
    wait(NULL); wait(NULL);
    return 0;
}
^O Help      ^O Write Out  ^M Where Is   ^K Cut        ^T Execute    ^C Location   ^U Undo       ^M-A Set Mark  ^M-J To Bracket ^M-Q Previous  ^B Back
^X Exit      ^R Read File  ^_ Replace    ^V Paste      ^J Justify    ^_ Go To Line ^M-E Redo     ^M-6 Copy     ^M-_ Where Was ^M-W Next     ^F Forward

Feb 26 2002 bashairyaqoob@HP: ~

bashairyaqoob@HP:~$ nano q5f.c
bashairyaqoob@HP:~$ gcc q5f.c -o q5f
bashairyaqoob@HP:~$ ./q5f
I am Process A and my PID is 5725
I am Process C and my PID is 5727
I am Process B and my PID is 5726
I am Process E and my PID is 5728
I am Process D and my PID is 5730
I am Process F and my PID is 5729
bashairyaqoob@HP:~$
```

(G):

```
GNU nano 7.2 q5g.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t B, C, D, E, F, G;
    printf("I am Process A and my PID is %d\n", getpid());
    B = fork();
    if(B == 0) {
        printf("I am Process B and my PID is %d\n", getpid());
        E = fork();
        if(E == 0) {
            printf("I am Process E and my PID is %d\n", getpid());
            exit(0);
        }
        wait(NULL); exit(0);
    }
    C = fork();
    if(C == 0) {
        printf("I am Process C and my PID is %d\n", getpid());
        G = fork();
        if(G == 0) {
            printf("I am Process G and my PID is %d\n", getpid());
            exit(0);
        }
        F = fork();
        if(F == 0) {
            printf("I am Process F and my PID is %d\n", getpid()); exit(0);
        }
        wait(NULL); wait(NULL);
        exit(0);
    }
    D = fork();
    if(D == 0) {
        printf("I am Process B and my PID is %d\n", getpid());
        exit(0);
    }
    wait(NULL); wait(NULL); wait(NULL);
    return 0;
}
```

```
bashairyaqoob@HP:~$ nano q5g.c
bashairyaqoob@HP:~$ gcc q5g.c -o q5g
bashairyaqoob@HP:~$ ./q5g
I am Process A and my PID is 5908
I am Process B and my PID is 5909
I am Process C and my PID is 5910
I am Process E and my PID is 5911
I am Process B and my PID is 5912
I am Process G and my PID is 5913
I am Process F and my PID is 5914
bashairyaqoob@HP:~$
```



(H):

```
GNU nano 7.2 q5g.c *
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t B, C, D, E, F, G, H;
    printf("I am Process A and my PID is %d\n", getpid());
    B = fork();
    if(B == 0) {
        printf("I am Process B and my PID is %d\n", getpid());
        E = fork();
        if(E == 0) {
            printf("I am Process E and my PID is %d\n", getpid());
            exit(0);
        }
        F = fork();
        if(F == 0) {
            printf("I am Process F and my PID is %d\n", getpid()); exit(0);
        }
        wait(NULL); wait(NULL); exit(0);
    }
    C = fork();
    if(C == 0) {
        printf("I am Process C and my PID is %d\n", getpid());
        exit(0);
    }
    D = fork();
    if(D==0){
        printf("I am Process D and my PID is %d\n", getpid());
        G = fork();
        if(G == 0) {
            printf("I am Process G and my PID is %d\n", getpid());
            exit(0);
        }
        H = fork();
        if(H == 0) {
            printf("I am Process H and my PID is %d\n", getpid()); exit(0);
        }
        wait(NULL); wait(NULL); exit(0);
    }
    wait(NULL); wait(NULL); wait(NULL);
}
```

```
bashairyaqoob@HP:~$ nano q5h.c
bashairyaqoob@HP:~$ gcc q5h.c -o q5h
bashairyaqoob@HP:~$ ./q5h
I am Process A and my PID is 6107
I am Process D and my PID is 6110
I am Process G and my PID is 6111
I am Process H and my PID is 6112
I am Process C and my PID is 6109
I am Process B and my PID is 6108
I am Process E and my PID is 6113
I am Process F and my PID is 6114
bashairyaqoob@HP:~$
```