Naming the file: 20BCE1798_EX-5_PROCESSES
Reg No: **20BCE1798**
Name: **ANSH GOEL**

Course Code: CSE2005
Course Name: Operating Systems (Embedded Lab)

Slot: L27+L28

Ex No 5: PROCESSES
Date: 11-02-22

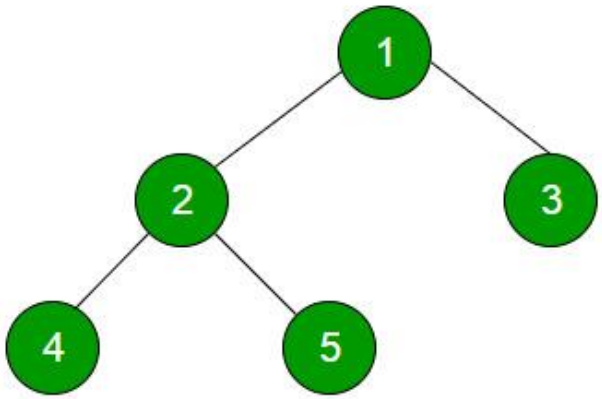
QUESTION:

Ex No 5

Date: 11-02-21

Processes

Create processes 1,2,3,4, and 5 as specified in the tree.



Print I am 1 from process 1, I am 2 from process 2..... Till process 5.

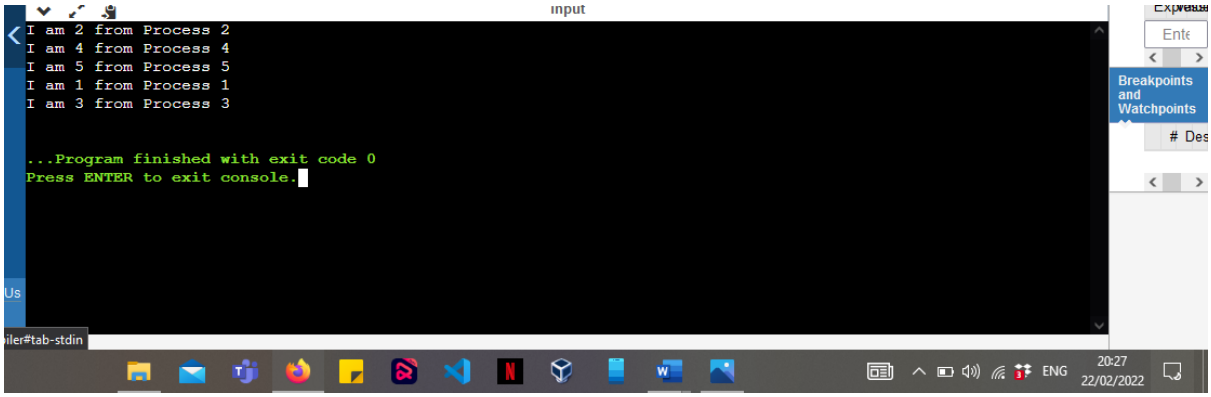
CODE FOR THE ABOVE QUESTION:

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
int
main ()
{
  pid_t p1, p2, p3, p4;
  p1 = fork ();
  if (p1 < 0)
    printf ("Fork Failure\n");
  else if (p1 == 0)
    {
      printf ("I am 2 from Process 2\n");
      p3 = fork ();
      if (p3 < 0)
    printf ("Fork Failure\n");
      else if (p3 > 0)
    {
      wait (NULL);
      //printf("I am Process 2\n");
      p4 = fork ();
      if (p4 < 0)
        printf ("Fork Failure");
      else if (p4 > 0)
        {
          wait (NULL);
          //printf("I am Process 2\n");
        }
       else if (p4 == 0)
         printf ("I am 5 from Process 5\n");
    }
      else if (p3 == 0)
    printf ("I am 4 from Process 4\n");
    }
   else if (p1 > 0)
    {
      wait (NULL);
      printf ("I am 1 from Process 1\n");
      p2 = fork ();
      if (p2 < 0)
    printf ("Fork Failure\n");
      else if (p2 > 0)
    {
      wait (NULL);
      //printf("I am Process 1\n");
    }
      else if (p2 == 0)
    printf ("I am 3 from Process 3\n");
    }
}
```

NAME – ANSH GOEL REG NO. – 20BCE1798

OUTPUT:



NAME – ANSH GOEL REG NO. – 20BCE1798

I took some reference form this code:

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
int
main ()
{
  pid_t p1, p2, p3, p4;
  p1 = fork ();
  if (p1 < 0)
    printf ("Fork Failure\n");
  else if (p1 == 0)
    {
      printf ("Child Process %d Parent %d\n", getpid (), getppid ());
      p3 = fork ();
      if (p3 < 0)
    printf ("Fork Failure\n");
      else if (p3 > 0)
    {
      wait (NULL);
      printf ("Parent Process %d\n", getpid ());
      p4 = fork ();
      if (p4 < 0)
        printf ("Fork Failure");
      else if (p4 > 0)
        {
          wait (NULL);
          printf ("Parent Process %d\n", getpid ());
        }
      else if (p4 == 0)
        printf ("Child Process %d Parent %d\n", getpid (), getppid ());
    }
      else if (p3 == 0)
    printf ("Child Process %d Parent %d\n", getpid (), getppid ());
    }
  else if (p1 > 0)
    {
      wait (NULL);
      printf ("Parent Process %d\n", getpid ());
      p2 = fork ();
      if (p2 < 0)
    printf ("Fork Failure");
      else if (p2 > 0)
    {
      wait (NULL);
      printf ("Parent Process %d\n", getpid ());
    }
      else if (p2 == 0)
    printf ("Child Process %d Parent %d\n", getpid (), getppid ());
    }
}
```