PROCESS:

1]Write a C program that uses fork system call to create a child process and analyze how child processes are created and also find how many child created with N continues fork calls. Make a study on use of global and local variables.

Solution:

```
#include<stdio.h>
#include<sys/types.h>
int main()
{
   int n,i;
   printf("Enter the number of file process to be created:");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
   {
          pid_t p;
          p = fork();
          if (p==-1)
          {
                   perror("error statement\n");
          }
          else if(p==0)
          {
                   printf("\nHello from child %d\n",i);
          }
          else if(p >0)
          {
                   printf("\nHello from the parent %d\n",i);
          }
          else
          {
                   printf("\nunable to create child\n");
          }
   }
}
```

2]Write a C program to demonstrate ORPHAN state of a process.

Solution:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
   pid_t p;
   p = fork();
   if(p==0)
   {
          printf("CHILD PROCESS:\n");
          printf("\nprocess_id:%d\n",getpid());
          printf("\nParent_id:%d\n",getppid());
          sleep(10);
          printf("CHILD PROCESS:\n");
          printf("\nprocess_id:%d",getpid());
          printf("\nparent_id:%d",getppid());
   }
   else if(p>0)
   {
          printf("PARENT PROCESS:\n");
          printf("process_id:%d",getpid());
   }
   else
   {
          printf("Cannot create child");
   }

}
```

3]Write a C program to demonstrate ZOMBIE state of a process.

Solution;

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<stdlib.h>
int main()
{
   pid_t p;
   p = fork();
   if(p==0)
   {
         exit(0);
   }
   else if(p>0)
   {
         sleep(5);
   }
   else
   {
         printf("unable to create child");
   }
}
```

4]Write a C program to create a child process and return the exit status to the parent. In parent get the exit status of the child and print.

Solution:

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<sys/wait.h>
#include<sys/types.h>
int main()
{
   pid_t p;
   p = fork();
   if(p==0)
   {
         exit(102);
   }
   int status;
   waitpid(p,&status,0);
   if(WIFEXITED(status))
   {
         int exit_status = WEXITSTATUS(status);
         printf("exit status:%d",exit_status);
   }
   else
   {
         printf("Unable to create child");
   }
}
```

5]Write a C program to create a child process that executes another C program that calculates the factorial of the given number using exec system call.

Factorial program:

```
#include<stdio.h>
int factorial(int n)
{
   int result =1;
   if(n<0)
   {
         printf("Enter the number that is n0n-negative");
   }
   for (int i = 2;i<=n;i++)
   {
         result *= i;
   }
   return result;
}
int main()
{
   int n,ans;
   printf("Enter the number:");
   scanf("%d",&n);
   ans = factorial(n);
   printf("Factorial is:%d",ans);
   return 0;
}
```

[note: after this run the command gcc factorial.c -o factorial.out]

Parent program:

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

```
int main()
{
   pid_t p;
   p = fork();
   int number;
   if(p<0)
   {
         perror("Error statement");
   }
   else if (p == 0)
   {
         printf("Enter the number:");
         scanf("%d",&number);
         if(number <0)
         {
                printf("Enter the non negative number");
                return 1;
         }
         char num_args[10];
         snprintf(num_args,sizeof(num_args),"%d",number);
         char *args[] = {"factorial.out",num_args,NULL};
         execvp("./factorial.out",args);
         fprintf(stderr,"Exec failed");
   }
   else
   {
         wait(NULL);
         printf("child completed");
   }
   return 0;
}
```

6]Write a C program to create a child process. The parent will pass an integer through pipe to the child. Child will calculate the factorial and return through the same pipe to parent. Then the parent will print the result.

Solution

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int factorial(int n)
{
   int result = 1;
   for(int i=2;i<=n;i++)
   {
         result *= i;
   }
   return result;
}
int main()
{
   int pipefd[2];
   int result,number;
   if(pipe(pipefd)==-1)
   {
         perror("Pipe");
         exit(EXIT_FAILURE);
   }

   pid_t p;
   p =fork();
   if(p<0)
   {
         perror("FORK");
         exit(EXIT_FAILURE);
   }
```

```
    else if(p == 0)
    {
        read(pipefd[0],&number,sizeof(number));
        result = factorial(number);
        write(pipefd[1],&result,sizeof(result));
        //exit(EXIT_SUCESS);
    }
    else
    {
        printf("Enter the number:");
        scanf("%d",&number);
        write(pipefd[1],&number,sizeof(number));
        wait(NULL);
        read(pipefd[0],&result,sizeof(result));
        printf("RESULT:%d",result);
    }
}
```

MATRIX MULTIPLICATION