Jayant Gaur
20194152
CS-7B

DISTRIBUTED SYSTEMS LAB
ASSIGNMENT 1

Q1. Write a program to create two processes. First process takes a string and passes it to second process through a pipe. The second process concatenates the received string with another string without using string function and sends it back to the first process for printing.

Solution:

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

int main()
{
    // We use two pipes
    // First pipe to send input string from parent
    // Second pipe to send concatenated string from child

    int fd1[2];  // Used to store two ends of first pipe
    int fd2[2];  // Used to store two ends of second pipe

    char fixed_str[] = "forgeeks.org";
    char input_str[100];
    pid_t p;

    if (pipe(fd1)==-1)
    {
        fprintf(stderr, "Pipe Failed" );
        return 1;
    }
    if (pipe(fd2)==-1)
    {
        fprintf(stderr, "Pipe Failed" );
        return 1;
    }

    scanf("%s", input_str);
    p = fork();

    if (p < 0)
```

```
    {
        fprintf(stderr, "fork Failed" );
        return 1;
    }

    // Parent process
    else if (p > 0)
    {
        char concat_str[100];

        close(fd1[0]);  // Close reading end of first pipe

        // Write input string and close writing end of first
        // pipe.
        write(fd1[1], input_str, strlen(input_str)+1);
        close(fd1[1]);

        // Wait for child to send a string
        wait(NULL);

        close(fd2[1]); // Close writing end of second pipe

        // Read string from child, print it and close
        // reading end.
        read(fd2[0], concat_str, 100);
        printf("Concatenated string %s\n", concat_str);
        close(fd2[0]);
    }

    // child process
    else
    {
        close(fd1[1]);  // Close writing end of first pipe

        // Read a string using first pipe
        char concat_str[100];
        read(fd1[0], concat_str, 100);

        // Concatenate a fixed string with it
        int k = strlen(concat_str);
        int i;
        for (i=0; i<strlen(fixed_str); i++)
            concat_str[k++] = fixed_str[i];

        concat_str[k] = '\0';   // string ends with '\0'
```

```
        // Close both reading ends
        close(fd1[0]);
        close(fd2[0]);

        // Write concatenated string and close writing end
        write(fd2[1], concat_str, strlen(concat_str)+1);
        close(fd2[1]);

        exit(0);
    }
}
```

Q2. **Develop a program in which the parent process sends two matrices to its child process through a pipe and the child process returns the sum of the matrices to the parent through a pipe. The parent should print the result.**

Solution:

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

int main()
{
        int fd[2];
        pipe(fd);
        printf("Pipe created successfully\n");
        int id = fork();
        int a[3][3];
        int b[3][3];

        if(id < 0)
        {
                printf("Fork failed\n");
                return 0;
        }

        if(id == 0)
        {
                //Child Process
            read(fd[0],a,9*sizeof(int));
```

```
            read(fd[0],b,9*sizeof(int));

           for(int i=0;i<3;i++)
             for(int j=0;j<3;j++)
                 a[i][j] +=b[i][j];

          write(fd[1],a,9*sizeof(int));
          close(fd[1]);

        }
        else
        {
                //Partent
                printf("Enter 1 matrix\n");
                for(int i=0;i<3;i++)
                   for(int j=0;j<3;j++)
                           scanf("%d",&a[i][j]);
             printf("Enter 2 matrix\n");
                for(int i=0;i<3;i++)
                        for(int j=0;j<3;j++)
                                scanf("%d",&b[i][j]);


                write(fd[1] , a , 9*(sizeof(int)));
                sleep(1);
                write(fd[1] , b, 9*sizeof(int));
                close(fd[1]);
                wait(NULL);

                read(fd[0] , a, 9*sizeof(int));
               for(int i=0;i<3;i++)
                  {
                   for(int j=0;j<3;j++)
                      printf("%d ",a[i][j]);
                 printf("\n");
                  }
                printf("Exiting Parent process");

        }
        return 0;
}
```