Task:Create a program where the parent process accepts a number from the user and sends it to the child process. In the child process, each digit of the number is separated and sent back to the parent process through separate pipes. After receiving all digits, the parent process calculates and displays the product of all the digits.

```
student@VW:~$ gedit inlab6q.c
[10]+  Killed                  gedit inlab6q.c
student@VW:~$ gcc inlab6q.c -o out
student@VW:~$ ./out
Enter a Number(Max 3 digits):456
Product of digits: 120
student@VW:~$
```

```
student@VW:~$ gcc inlab6q.c -o out
student@VW:~$ ./out
Enter a Number(Max 3 digits):231
Product of digits: 6
student@VW:~$ ./out
Enter a Number(Max 3 digits):341
Product of digits: 12
student@VW:~$ ./out
Enter a Number(Max 3 digits):125
Product of digits: 10
student@VW:~$
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
void numseparate(int num, int pipefd[3][2]) {
int i = 0;
int digits[3];
while (num>0 && i<3) {
digits[i] = num % 10;
num /= 10;
i++;
}

for (int j = 0;j<i;j++) {
write(pipefd[j][1],&digits[j], sizeof(int));

close(pipefd[j][1]);
}
}
int main() {
int num;
printf("Enter a Number(Max 3 digits):");
scanf("%d",&num);
int pipefd[3][2];
for (int i = 0;i<3; i++) {
if (pipe(pipefd[i]) == -1) {
perror("Pipe failed.");
return 1;
}
}
pid_t p1 = fork();
if (p1>0) {
wait(NULL);
int product = 1, digit;
for (int i = 0;i<3; i++) {
close(pipefd[i][1]);
if (read(pipefd[i][0],&digit, sizeof(int))> 0) {
product *= digit;
}
close(pipefd[i][0]);
}
printf("Product of digits: %d\n", product);
}
else if (p1 == 0) {
numseparate(num, pipefd);
exit(0);
}
else {
printf("Fork failed\n");
return 1;
}
}
```

Producer.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#define FIFO_FILE "/tmp/myfifo"

int main() {
    int fd;
    char buffer[BUFSIZ];
    ssize_t num_bytes;
    mkfifo(FIFO_FILE, 0777);
    fd = open(FIFO_FILE, O_WRONLY);

    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    while (1) {
        printf("Producer: Enter a message (or 'exit' to quit): ");
        fgets(buffer, BUFSIZ, stdin);

        num_bytes = write(fd, buffer, strlen(buffer));

        if (num_bytes == -1) {
            perror("write");
            exit(EXIT_FAILURE);
        }

        if (strncmp(buffer, "exit", 4) == 0) {
            break;
        }
    }
    close(fd);
    unlink(FIFO_FILE);
    return 0;
}
```

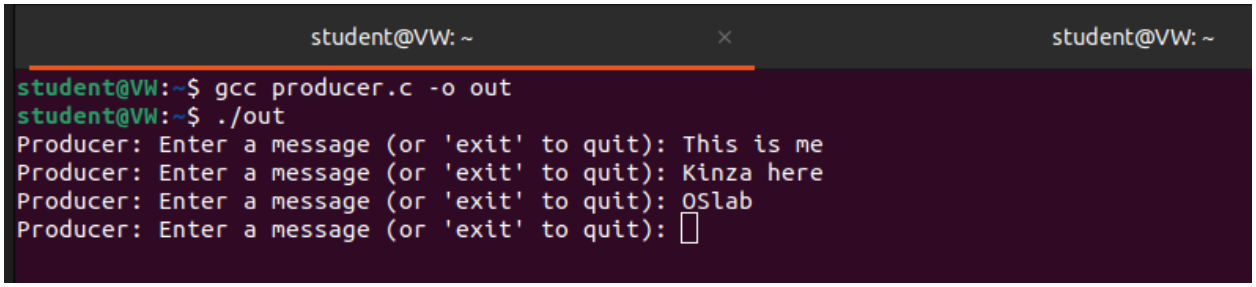
Consumer.c

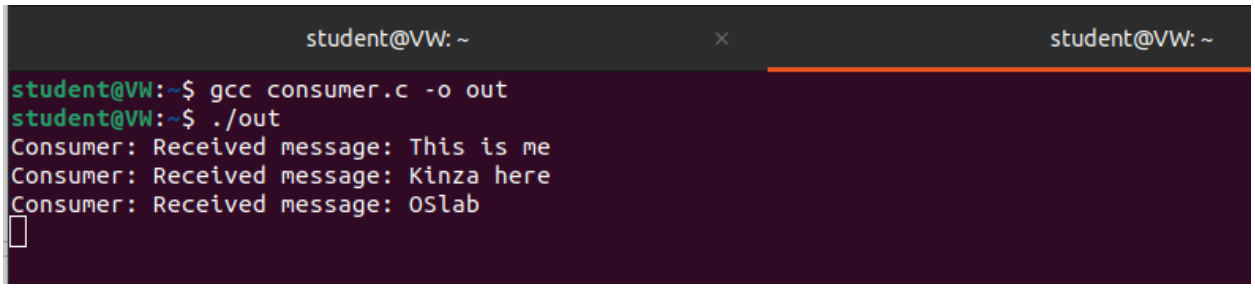
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>

#define FIFO_FILE "/tmp/myfifo"

int main() {
    int fd;
    char buffer[BUFSIZ];
    ssize_t num_bytes;
    fd = open(FIFO_FILE, O_RDONLY);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }
    while (1) {
        num_bytes = read(fd, buffer, BUFSIZ);
        if (num_bytes == -1) {
            perror("read");
            exit(EXIT_FAILURE);
        }
        if (num_bytes == 0) {
            continue;
        }
        buffer[num_bytes] = '\0';
        printf("Consumer: Received message: %s", buffer);
        if (strncmp(buffer, "exit", 4) == 0) {
            break;
        }
    }
    close(fd);
    return 0;
}
```

Output:


student@VW: ~
student@VW: ~
student@VW:~$ gcc producer.c -o out
student@VW:~$ ./out
Producer: Enter a message (or 'exit' to quit): This is me
Producer: Enter a message (or 'exit' to quit): Kinza here
Producer: Enter a message (or 'exit' to quit): OSlab
Producer: Enter a message (or 'exit' to quit):


student@VW: ~
student@VW: ~
student@VW:~$ gcc consumer.c -o out
student@VW:~$ ./out
Consumer: Received message: This is me
Consumer: Received message: Kinza here
Consumer: Received message: OSlab