OS LAB WEEK5

Name :Sagnik Chatterjee
REg :180905478
Section :B
ROll no :61

Q1.
Code:
Q1_consumer.c

```
/*
AUTHOR :SAGNIK CHATTERJEE
DATE : DEC 15,2020
USAGE : ./q1c

*/

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

#define FIFO_NAME "/tmp/my_fifo"
#define BUFFER_SIZE PIPE_BUF

int main()
{
    int pipe_fd;
    int res;
    int open_mode=O_RDONLY;
    char buffer[4];
```

```
    int bytes_read=0;

    memset(buffer,'\0',sizeof(buffer));
    printf("[STATUS] Opening FIFO O_RDONLY\n");
    pipe_fd=open(FIFO_NAME,open_mode);
    printf("[STATUS] Pipefd result :- %d \n",pipe_fd);

    if (pipe_fd!=-1)
    {
        for(int i=0;i<4;i++)
        {   //printing the 4 integers to the fifo queue
            res=read(pipe_fd,buffer,BUFFER_SIZE);
            if(res==-1){
                printf("[ERROR] Read error on pipe.\n");
                exit(1);
            }
            printf("%d\n",atoi(buffer));
            bytes_read+=res;
            buffer[0]='\n';//clear the buffer

        }
        (void)close(pipe_fd);//close the filedescriptor
    }
    else{
        fprintf(stderr,"[ERROR] File could not be opened.\n");
        exit(EXIT_FAILURE);
    }


    printf("[STATUS]  Finished and %d bytes read \n",bytes_read);
    exit(EXIT_SUCCESS);
}
```

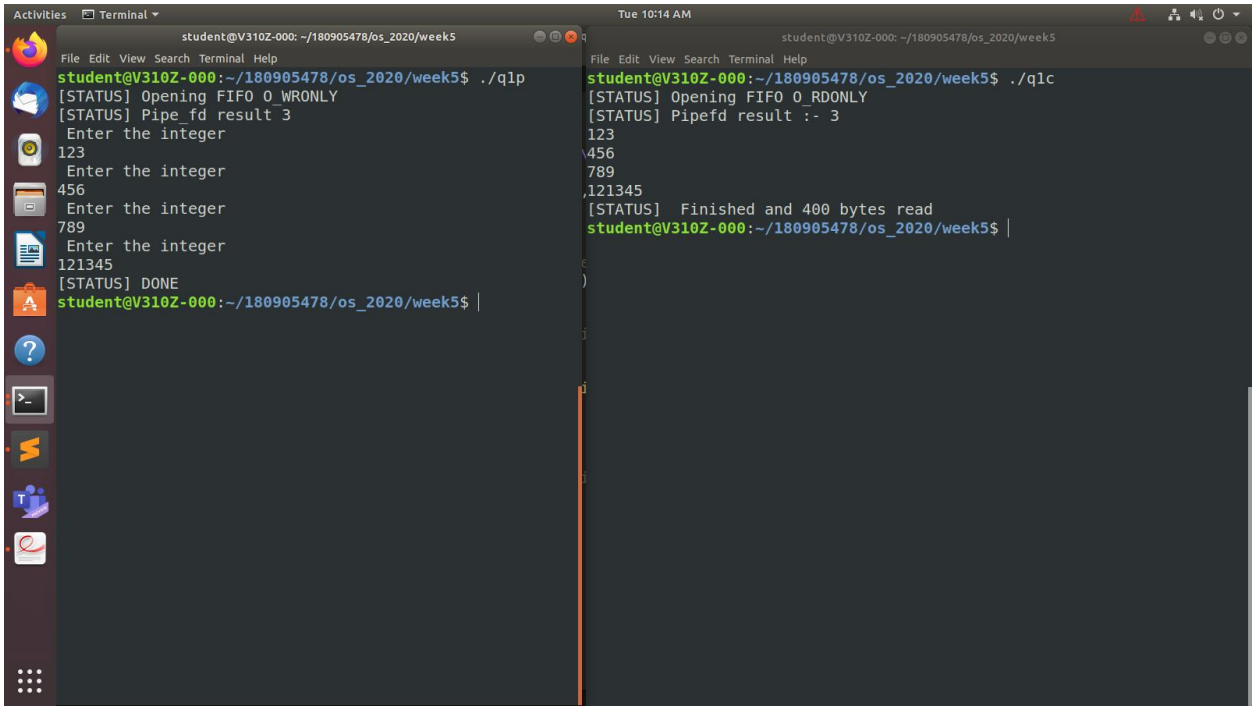
Q1_producer.c

```
/*
```

```
AUTHOR :SAGNIK CHATTERJEE
DATE : DEC 15,2020
USAGE : ./q1p

*/

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

#define FIFO_NAME "/tmp/my_fifo"
#define BUFFER_SIZE PIPE_BUF

int main()
{
    int pipe_fd;
    int res;
    int open_mode=O_WRONLY;
    int bytes_sent=0;
    char buffer[100];

    if (access(FIFO_NAME,F_OK)==-1){
        res=mkfifo(FIFO_NAME,0777);

        if (res!=0)
        {
            fprintf(stderr,"[ERROR] Couldn't create fifo %s\n",FIFO_NAME
);
            exit(EXIT_FAILURE);
        }
```

```
    }
    printf("[STATUS] Opening FIFO O_WRONLY\n");
    pipe_fd=open(FIFO_NAME,open_mode);
    printf("[STATUS] Pipe_fd result %d \n",pipe_fd);
    if (pipe_fd!=-1)
    {
        for (int i=0;i<4;i++)
        {   //writing the 4 integers in the fifo queue
            printf(" Enter the integer \n");
            scanf("%s",buffer);
            res=write(pipe_fd,buffer,100);
            //buffer[0]='\n';//clear the pipe
            if (res==-1)
            {
                fprintf(stderr,"[ERROR] Write error on pipe\n");
                exit(EXIT_FAILURE);
            }
            bytes_sent+=res;
        }
        (void)close(pipe_fd);//close the file descriptor
    }
    else
    {
        printf("[ERROR] Couldn't read from the pipe file descriptor.\n");
        exit(EXIT_FAILURE);
    }
    printf("[STATUS] DONE \n");
    exit(EXIT_SUCCESS);
}
```

Screenshot :



Q2

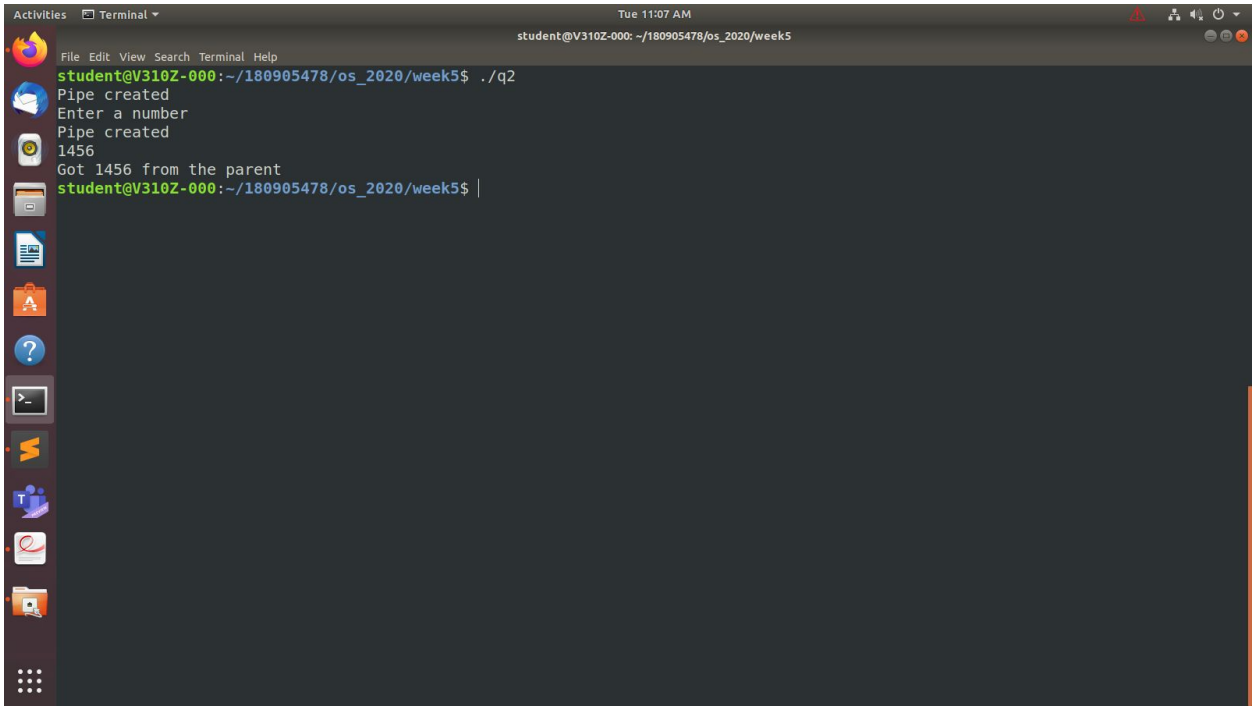
Code:

```
/*
AUTHOR :SAGNIK CHATTERJEE
DATE : DEC 15,2020
USAGE : ./q2

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

```
int main(){
        int pfd[2];
        pid_t cpid;
        int buff;

        if(pipe(pfd)==-1){
        perror("[STATUS] Pipe failure\n");
        exit(EXIT_FAILURE);
        }

        cpid = fork();
        if(cpid==-1){
        perror("[STATUS] Fork error\n");
        exit(EXIT_FAILURE);
        }
        else{
        printf("[STATUS] Pipe created\n");
        }
        if(cpid==0){
        //child process reads from pipe
        close(pfd[1]);
        int y;
        read(pfd[0],&y,sizeof(int));
        close(pfd[0]);
        printf("[STATUS] Got %d from the parent\n",y);
        }
        else{
        //parent writes to child
        close(pfd[0]);
        printf("[STATUS] Enter a number\n");
        scanf("%d",&buff);
        write(pfd[1],&buff,sizeof(int));
        close(pfd[1]);
        wait(NULL);
        exit(EXIT_SUCCESS);
```

```
        }

}
```

Screenshot:



Q3

Code:

Part1 : /first writing and then reading

```
/*
AUTHOR :SAGNIK CHATTERJEE
DATE : DEC 15,2020
USAGE : ./q3_p1

*/
```

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include<fcntl.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/stat.h>
#include<limits.h>
#define FILE_NAME "/tmp/my_fifo"

int main()
{
    int pipe_fd;
    int res;
    char buffer[1024];

    memset(buffer,'\0',sizeof(buffer));

    //first writing and then reading

    if(access(FILE_NAME,F_OK)==-1)
    {
        res = mkfifo(FILE_NAME,0777);
        if(res!=0)
        {
            fprintf(stderr,"[ERROR] Couldn't create fifo %s\n",
FILE_NAME);
            exit(EXIT_FAILURE);
        }
    }

    printf("[STATUS] Opening FIFO_WRONLY");
    pipe_fd = open(FILE_NAME,O_WRONLY);
    printf("[INPUT] Input....\n");
```

```
    fgets(buffer,1024,stdin);
    if(write(pipe_fd,buffer,strlen(buffer)+1)==-1){
        fprintf(stderr,"[ERROR] Error in writing \n");
        exit(EXIT_FAILURE);
    }
    close(pipe_fd);

    printf("[STATUS] Opening FIFO_RDONLY\n");
    pipe_fd = open(FILE_NAME,O_RDONLY);

    if(pipe_fd==-1){
        fprintf(stderr,"[ERROR] Error in pie filedescriptor\n");
        exit(EXIT_FAILURE);
    }

    memset(buffer,'\0',sizeof(buffer));
    read(pipe_fd,buffer,1024);
    close(pipe_fd);

    printf("[STATUS] Output: Reads %s\n",buffer);
    return 0;


}
```

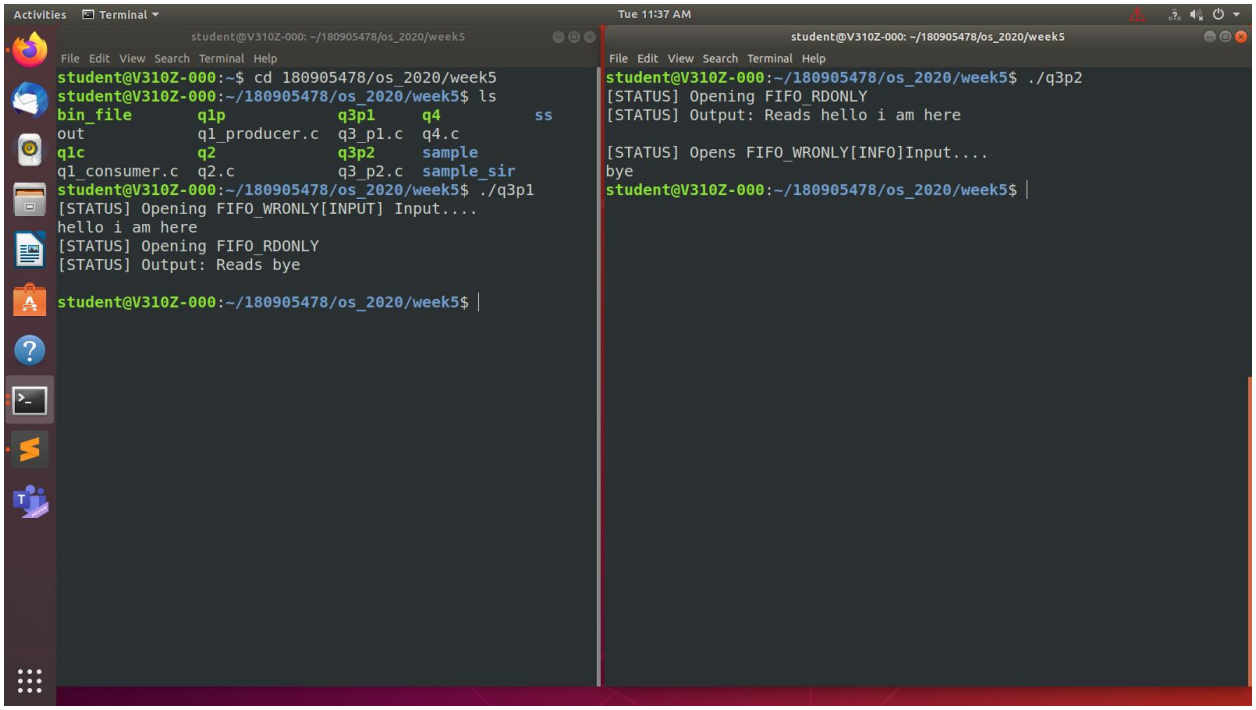
Part2 : //first reading and then writing

```
/*
AUTHOR :SAGNIK CHATTERJEE
DATE : DEC 15,2020
USAGE : ./q3_p2

*/
```

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

#define FILE_NAME "/tmp/my_fifo"


int main(){

    int pipe_fd;
    int res;
    char buffer[1024];


    //first reading and then writing

    if(access(FILE_NAME,F_OK)==-1)
    {
        res = mkfifo(FILE_NAME,0777);
        if(res!=0)
        {
            fprintf(stderr,"[ERROR] Couldn't create fifo %s\n",
FILE_NAME);
            exit(EXIT_FAILURE);
        }
    }


    printf("[STATUS] Opening FIFO_RDONLY\n");
    pipe_fd = open(FILE_NAME,O_RDONLY);
```

```
    memset(buffer,'\0',sizeof(buffer));
    if(read(pipe_fd,buffer,1024)==-1){
        fprintf(stderr,"[ERROR] READ error \n");
        exit(EXIT_FAILURE);
    }
    close(pipe_fd);

    printf("[STATUS] Output: Reads %s\n",buffer);

    printf("[STATUS] Opens FIFO_WRONLY");
    pipe_fd = open(FILE_NAME,O_WRONLY);
    printf("[INFO]Input....\n");
    memset(buffer,'\0',sizeof(buffer));
    fgets(buffer,1024,stdin);
    if(write(pipe_fd,buffer,strlen(buffer)+1)==-1){
        fprintf(stderr,"[ERROR] Error writing\n");
        exit(EXIT_FAILURE);
    }
    close(pipe_fd);

    return 0;
}
```

Screenshot :



Q4

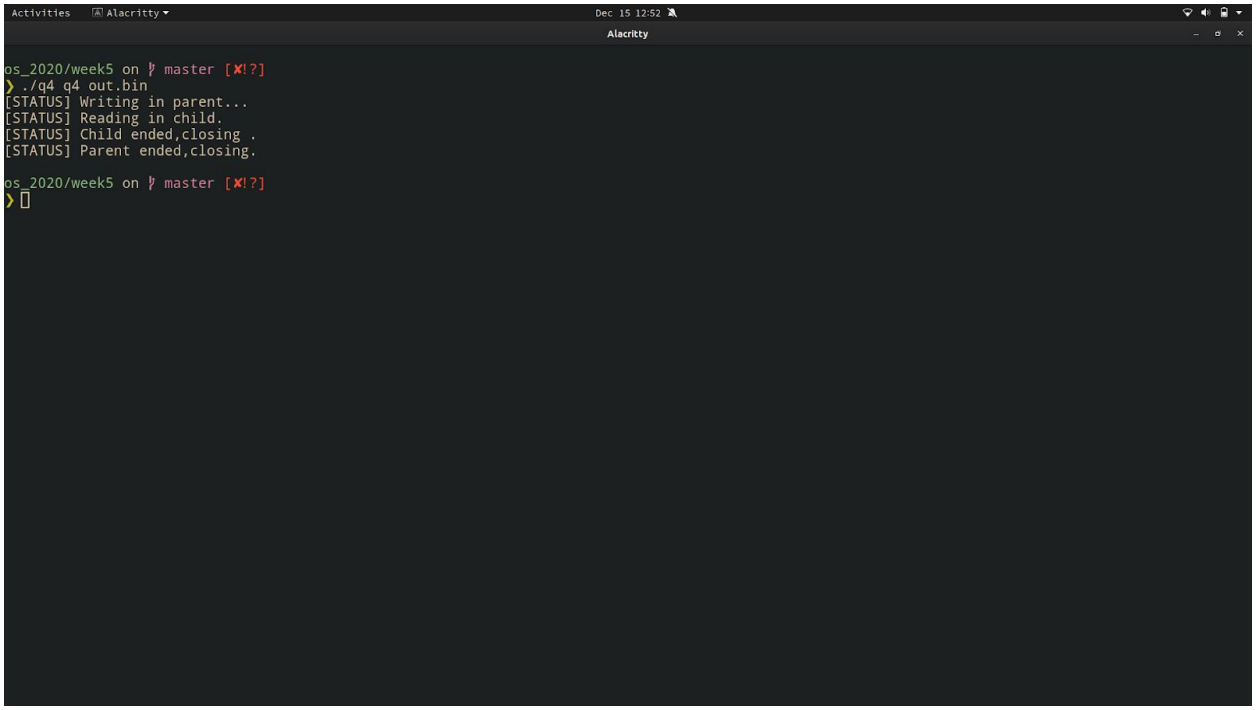
Code:

```
/*
AUTHOR :SAGNIK CHATTERJEE
DATE : DEC 15,2020
USAGE : ./q4 <input_file> <output_file>
*/




#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <assert.h>
```

```
#include <sys/wait.h>

int main(int argc, char **argv)
{
   int fd[2];
   pid_t pid;
   char buf[1024];
   if(argc!=3 ){
       printf("[ERROR] Usage : %s <inputfile>
<outputfile> \n",argv[0]);
       exit(EXIT_FAILURE);
   }
   if(pipe(fd)==-1){
       perror("[ERROR] Pipe Error\n");
       exit(EXIT_FAILURE);
   }
   pid = fork();
   if(pid==-1){
       perror("[ERROR] Fork Error\n");
       exit(EXIT_FAILURE);
   }
   else if(pid==0){
       //for child process write into binary file
       char ch;
       FILE *fw;
       fw = fopen(argv[2], "wb");
       if (fw == NULL){
           printf("[ERROR] Output binary file can't be
opened\n");
           exit(EXIT_FAILURE);
       }
       printf("[STATUS] Reading in child. \n");
```

```
        close(fd[1]); //close unused write end
        while(read(fd[0], &buf, strlen(buf))>0){
            fputs(buf, fw);
        }
        close(fd[0]);
        printf("[STATUS] Child ended,closing .\n");
        fclose(fw);
        exit(EXIT_SUCCESS);
    }
    else{
        //parent process read from the binary file
        char ch;
        FILE *fw = fopen(argv[1], "rb");
        if (fw == NULL){
            printf("[ERROR] Input binary file can't be
opened\n");
            exit(EXIT_FAILURE);
        }
        printf("[STATUS] Writing in parent...\n");
        close(fd[0]); //close unused read end
        while(fgets(buf, 1024, fw) !=NULL){
            write(fd[1], buf, strlen(buf));
        }
        close(fd[1]); //reader will see EOF
        wait(NULL); //wait for child to terminate
        printf("[STATUS] Parent ended,closing.\n");
        fclose(fw);
        exit(EXIT_SUCCESS);
    }
    return 0;
}
```

Screenshot :


Activities
Alacritty
Dec 15 12:52
Alacritty
os_2020/week5 on master [✘!?]
❯ ./q4 q4 out.bin
[STATUS] Writing in parent...
[STATUS] Reading in child.
[STATUS] Child ended,closing .
[STATUS] Parent ended,closing.
os_2020/week5 on master [✘!?]
❯