CONTENTS

ASSIGNMENT- 3

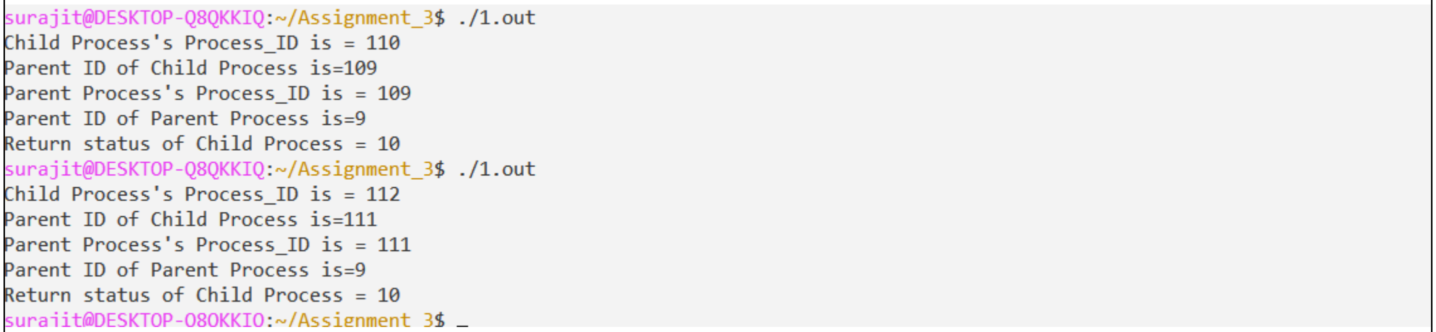
QUESTION -1

Write a C program to create a child process. The parent process must wait until the child finishes. Both the processes must print their own pid and parent pid. Additionally the parent process should print the exit status of the child.

CODE-

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
    int val=fork();
    int status;
    if(val==0)
    {
        //child process
        printf("\nProcess ID of child process is = %d\nParent Process ID of child process is
= %d\n\n",getpid(),getppid());
        return 7;
    }
    else
    {
        //parent process
        wait(&status);
        printf("Process ID of parent process is = %d\nParent Process ID of parent process is
= %d\n",getpid(),getppid());
        if(WIFEXITED(status))
        {
            printf("Return status of child process = %d\n",WEXITSTATUS(status));
        }
    }
    return 0;
}
```

OUTPUT SCREEN-



```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$ ./1.out
Child Process's Process_ID is = 110
Parent ID of Child Process is=109
Parent Process's Process_ID is = 109
Parent ID of Parent Process is=9
Return status of Child Process = 10
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$ ./1.out
Child Process's Process_ID is = 112
Parent ID of Child Process is=111
Parent Process's Process_ID is = 111
Parent ID of Parent Process is=9
Return status of Child Process = 10
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$ _
```

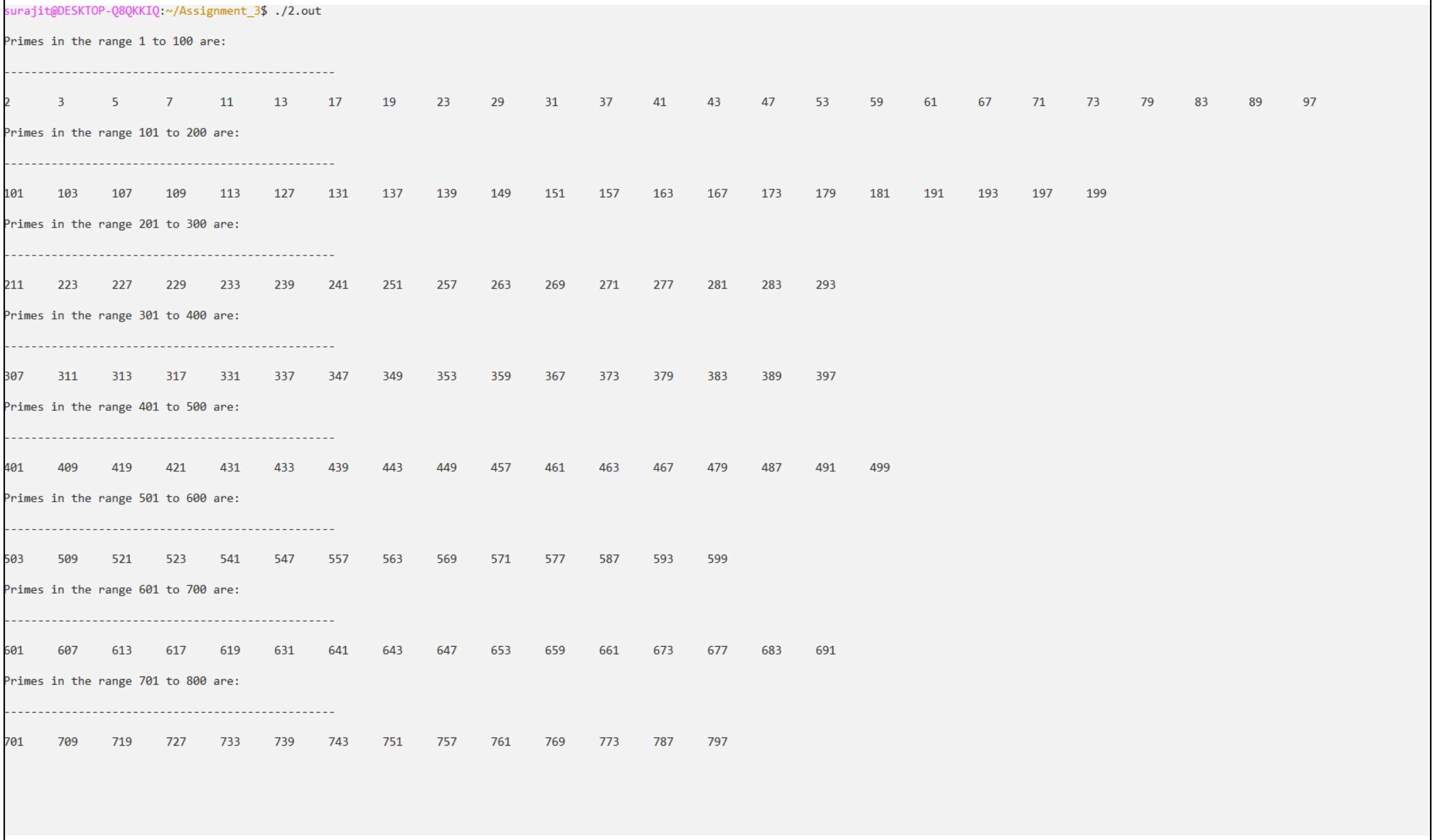
QUESTION -2

Write a C program which prints prime numbers between the range 1 to 10,00,000 by creating ten child processes and subdividing the task equally among all child processes, i.e., the first child should print prime numbers in the range 1 to 1,00,000, the second child in the range 1,00,001 to 2,00,000, ... The child processes must run in parallel and the parent process must wait until all the child processes finish.

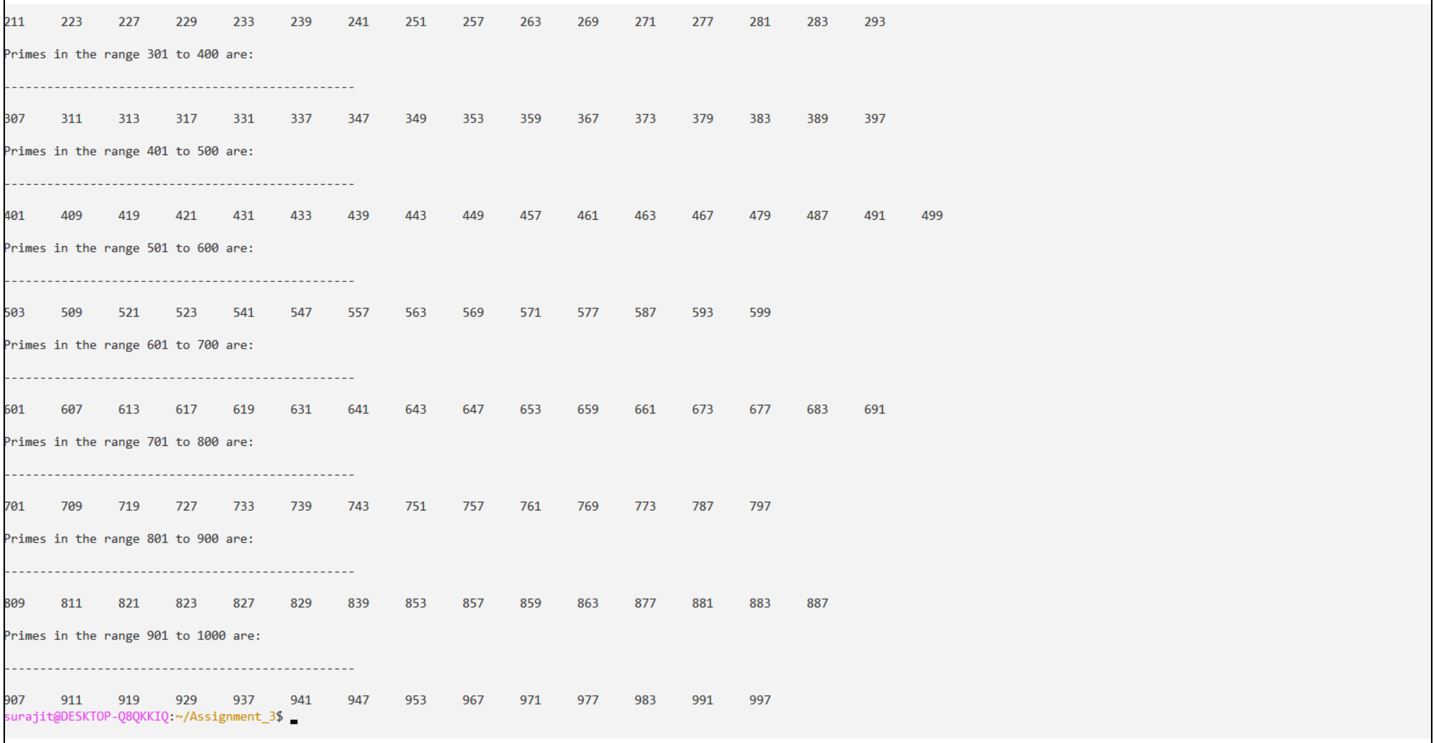
CODE-

```
// here the maximum limit is taken to be 1000 instead of 1000000
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<stdlib.h>
int is_prime(int n)
{
    int i;
    if(n==1)
    return 0;
    for(i=2;i*i<=n;i++)
    {
        if(n%i==0)
        return 0;
    }
    return 1;
}
void print_prime(int s,int e)
{
    int i;
    printf("\n\nPrime numbers between %d to %d are :\n",s,e);
    printf("______________________________________________________________\n");
    for(i=s;i<=e;i++)
    {
        if(is_prime(i))
        printf("%d\t",i);
    }
    printf("\n");
    return;
}
int main()
{
    int i,start,end,status;
    for(i=0;i<10;i++)
    {
        if(fork()==0)
        {
            start=100*i+1;
            end=start+99;
            print_prime(start,end);
            exit(0);
        }
          else{
                  wait(&status);
          }
    }
    printf("Parent terminates\n");
    return 0;
}
```

OUTPUT SCREEN-



```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$ ./2.out
Primes in the range 1 to 100 are:
------------------------------------------------
2      3      5      7      11     13     17     19     23     29     31     37     41     43     47     53     59     61     67     71     73     79     83     89     97
Primes in the range 101 to 200 are:
------------------------------------------------
101    103    107    109    113    127    131    137    139    149    151    157    163    167    173    179    181    191    193    197    199
Primes in the range 201 to 300 are:
------------------------------------------------
211    223    227    229    233    239    241    251    257    263    269    271    277    281    283    293
Primes in the range 301 to 400 are:
------------------------------------------------
307    311    313    317    331    337    347    349    353    359    367    373    379    383    389    397
Primes in the range 401 to 500 are:
------------------------------------------------
401    409    419    421    431    433    439    443    449    457    461    463    467    479    487    491    499
Primes in the range 501 to 600 are:
------------------------------------------------
503    509    521    523    541    547    557    563    569    571    577    587    593    599
Primes in the range 601 to 700 are:
------------------------------------------------
601    607    613    617    619    631    641    643    647    653    659    661    673    677    683    691
Primes in the range 701 to 800 are:
------------------------------------------------
701    709    719    727    733    739    743    751    757    761    769    773    787    797
```



```
211     223     227     229     233     239     241     251     257     263     269     271     277     281     283     293

Primes in the range 301 to 400 are:

-------------------------------------------------

307     311     313     317     331     337     347     349     353     359     367     373     379     383     389     397

Primes in the range 401 to 500 are:

-------------------------------------------------

401     409     419     421     431     433     439     443     449     457     461     463     467     479     487     491     499

Primes in the range 501 to 600 are:

-------------------------------------------------

503     509     521     523     541     547     557     563     569     571     577     587     593     599

Primes in the range 601 to 700 are:

-------------------------------------------------

601     607     613     617     619     631     641     643     647     653     659     661     673     677     683     691

Primes in the range 701 to 800 are:

-------------------------------------------------

701     709     719     727     733     739     743     751     757     761     769     773     787     797

Primes in the range 801 to 900 are:

-------------------------------------------------

809     811     821     823     827     829     839     853     857     859     863     877     881     883     887

Primes in the range 901 to 1000 are:

-------------------------------------------------

907     911     919     929     937     941     947     953     967     971     977     983     991     997
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$
```

QUESTION -3

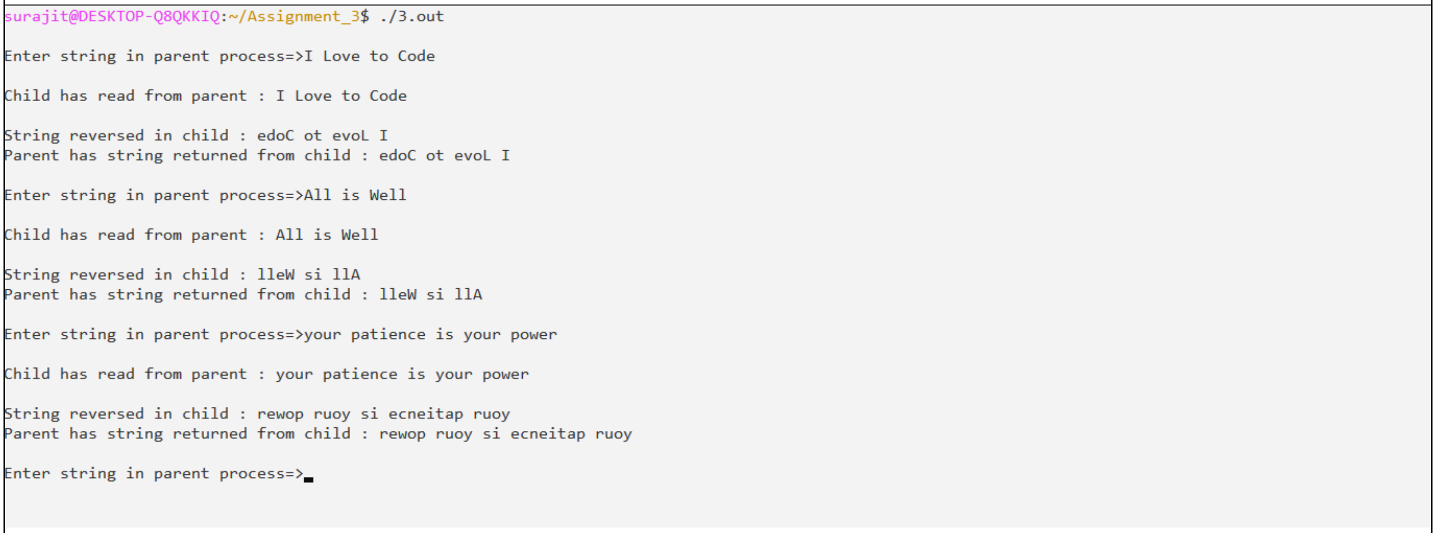
Write a C program which creates a child process and sends a string (input by user) which the child process reverses and sends it back to the parent. The IPC to be used is pipe. Both the processes terminate when the input string is "quit".

CODE-

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

int main(){
        int fd1[2], fd2[2];
        pipe(fd1);  //for parent write child read
        pipe(fd2);  //for child write parent read
        if(fork()!=0){
                //Parent Process
                //loop to repetitively input strings
                do{
                        char buffer[100]={0};
                        printf("\nEnter string in parent process=>");
                        scanf("\n");
                        scanf("%[^\n]",buffer);

                        //printf("\nLength %s in parent:%d",buffer,(int)strlen(buffer));

                        //if quit is entered then exit from parent process
                        if(strcmp("quit",buffer)==0){
                                exit(1);
                        }
                        //write and pass to child through pipe1
                        write(fd1[1], buffer, strlen(buffer));

                        //read child's string via pipe2
                        read(fd2[0], buffer, 100);
                        int len=strlen(buffer);
                        buffer[len]='\0';
                        printf("\nParent has string returned from child : %s\n", buffer);
                }while(1);
        }
        else{
                // Child Process
                //loop to repetitively reverse string from parent and sent it back to parent
                do{
                        char buffer[100]={0};
```

```
                    sleep(2);
                    //read parent's string via pipe1
                    read(fd1[0], buffer, 100);
                    int len=strlen(buffer);
                    buffer[len]='\0';
                    //printf("\nLength of %s in child:%d",buffer,len);
                    int j=0,k=len-1;
                    printf("\nChild has read from parent : %s\n", buffer);
                    //loop to reverse string from parent
                    while(j<k){
                          char ch=buffer[j];
                          buffer[j]=buffer[k];
                          buffer[k]=ch;
                          j++;
                          k--;
                   }
                   printf("\nString reversed in child : %s",buffer);
                   fflush(stdout);   //flushed standard output
                    //write and pass to parent ia pipe2
                    write(fd2[1], buffer, len);
             }while(1);
             //exit(0);
      }
      return 0;
}
```

OUTPUT SCREEN-



```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$ ./3.out

Enter string in parent process=>I Love to Code

Child has read from parent : I Love to Code

String reversed in child : edoC ot evoL I
Parent has string returned from child : edoC ot evoL I

Enter string in parent process=>All is Well

Child has read from parent : All is Well

String reversed in child : lleW si llA
Parent has string returned from child : lleW si llA

Enter string in parent process=>your patience is your power

Child has read from parent : your patience is your power

String reversed in child : rewop ruoy si ecneitap ruoy
Parent has string returned from child : rewop ruoy si ecneitap ruoy

Enter string in parent process=>
```

QUESTION -4

Write a C program which prints the following menu

1. ls

2. pwd

3. uname

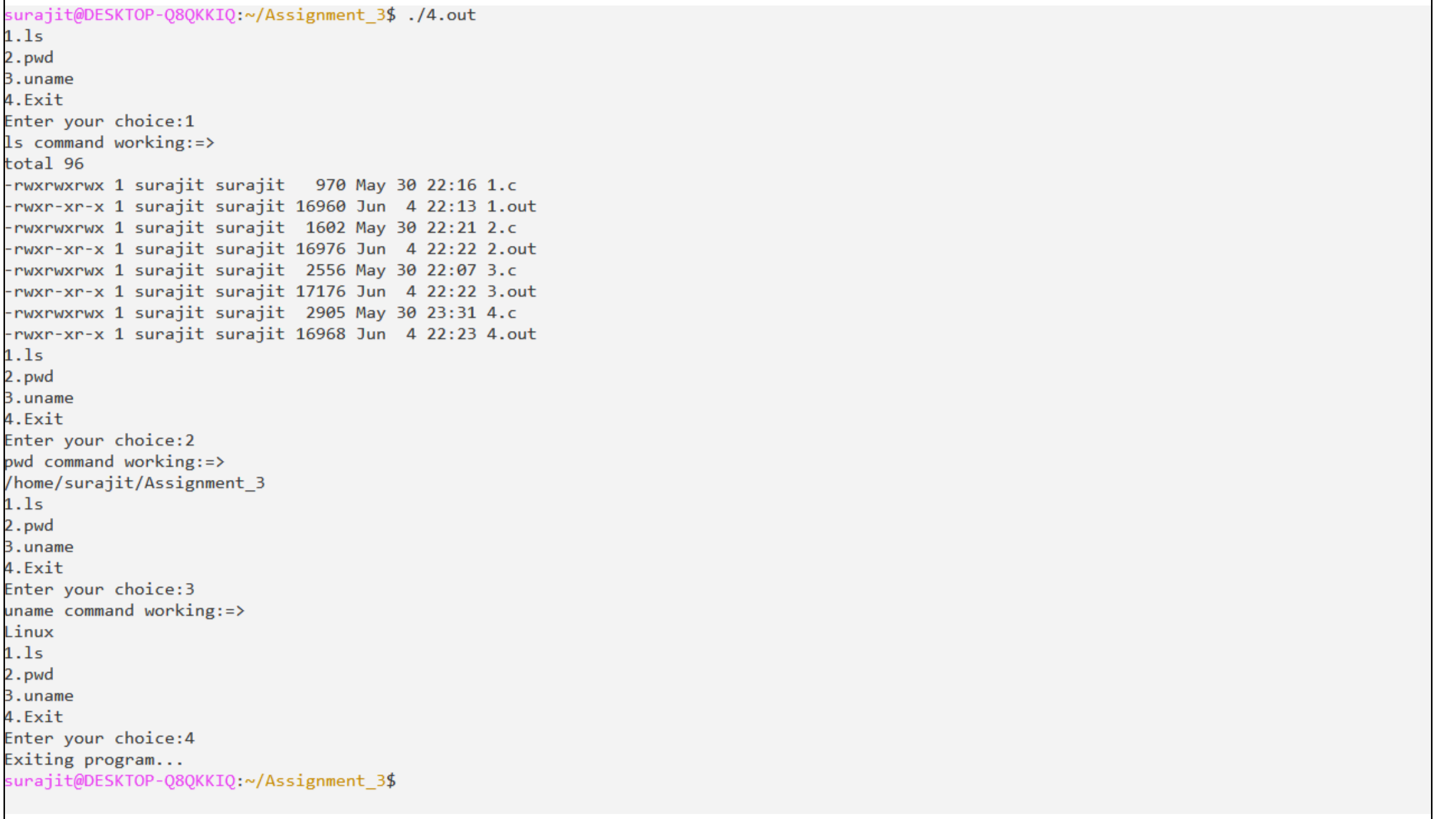
4. exit

When, the user provides an input, the parent process creates a child process [if user's choice is between 1-3] and executes the corresponding command [use execv() system call]. The main process waits for the child to finish and displays the menu again. The parent process terminates if user's choice is 4.

CODE-

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<stdlib.h>
#include<string.h>
int main()
{
   int op,status;
       char *str1[]={"/bin/pwd",NULL};
       char *str2[]={"/bin/ls","-l",NULL};
       char *str3[]={"/bin/uname",NULL};
   do
   {
      printf("\n******************* MENU ******************\n");
      printf("_________________________________________________\n");
      printf("\n1. ls\n2. pwd\n3. uname\n4. exit\nEnter your choice = ");
      scanf("%d",&op);
      switch(op)
      {
         case 1:
            printf("\nYou have selected ls command\n\n");
            if(fork()==0)
            {
            execv(str2[0],str2);
            }
            else
            wait(&status);
            break;
         case 2:
            printf("\nYou have selected pwd command\n\n");
            if(fork()==0)
            {
            execv(str1[0],str1);
            }
            else
            wait(&status);
            break;
        case 3:
            printf("\nYou have selected uname command\n\n");
            if(fork()==0)
            {
            execv(str3[0],str3);
            }
            else
            wait(&status);
            break;
        case 4:
            printf("Exiting from program\n");
            break;
        default:
            printf("Enter correct option between 1 to 4\n");
            break;
      }
  }while(op!=4);
  return 0;
}
```

OUTPUT SCREEN



```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$ ./4.out
1.ls
2.pwd
3.uname
4.Exit
Enter your choice:1
ls command working:=>
total 96
-rwxrwxrwx 1 surajit surajit   970 May 30 22:16 1.c
-rwxr-xr-x 1 surajit surajit 16960 Jun  4 22:13 1.out
-rwxrwxrwx 1 surajit surajit  1602 May 30 22:21 2.c
-rwxr-xr-x 1 surajit surajit 16976 Jun  4 22:22 2.out
-rwxrwxrwx 1 surajit surajit  2556 May 30 22:07 3.c
-rwxr-xr-x 1 surajit surajit 17176 Jun  4 22:22 3.out
-rwxrwxrwx 1 surajit surajit  2905 May 30 23:31 4.c
-rwxr-xr-x 1 surajit surajit 16968 Jun  4 22:23 4.out
1.ls
2.pwd
3.uname
4.Exit
Enter your choice:2
pwd command working:=>
/home/surajit/Assignment_3
1.ls
2.pwd
3.uname
4.Exit
Enter your choice:3
uname command working:=>
Linux
1.ls
2.pwd
3.uname
4.Exit
Enter your choice:4
Exiting program...
surajit@DESKTOP-Q8QKKIQ:~/Assignment_3$
```

ASSIGNMENT- 4

QUESTION -1

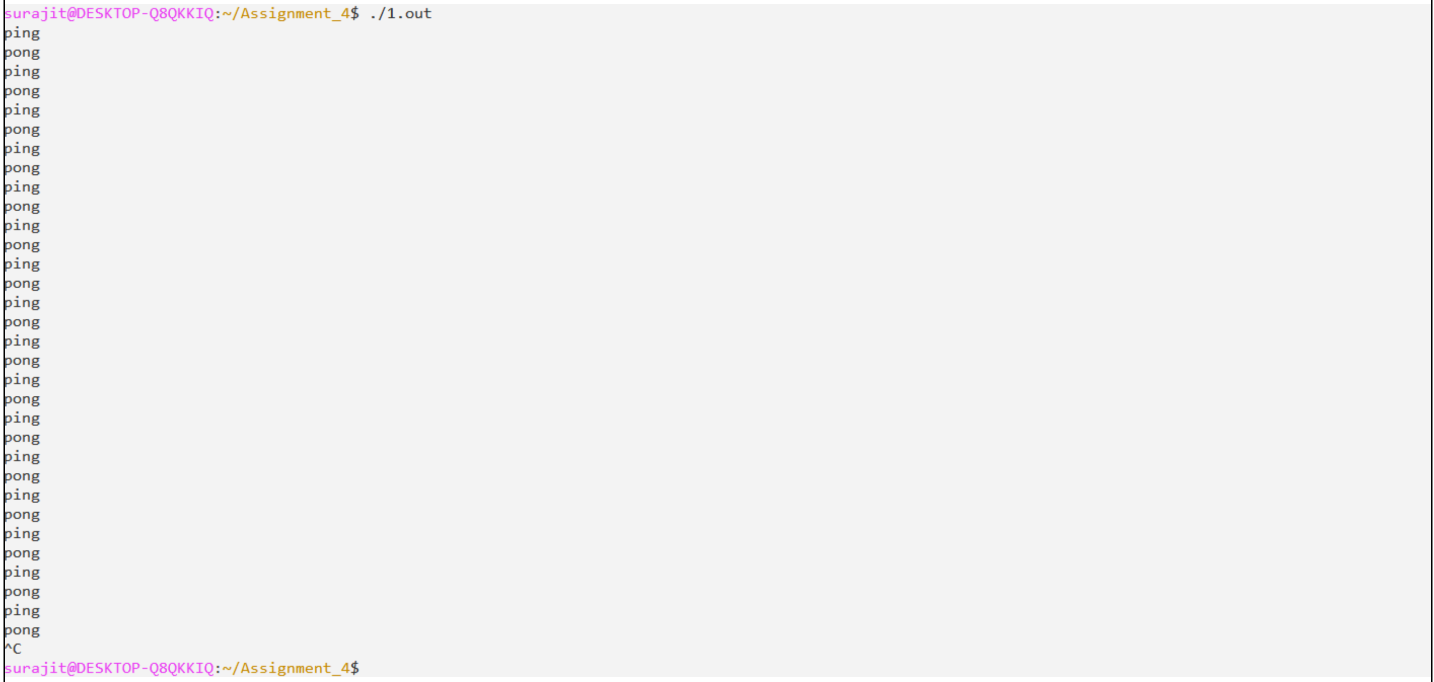
1. Ping and Pong are two separate processes executing their respective tasks. They should synchronize among themselves using a shared variable turn initialized to 0, such that they forever take turns, alternately printing “ping” and “pong”.

```
/∗ Ping p r o c e s s ∗/
while ( t r u e ){
while ( tu rn != 0 );
p r i n t ( ” pin g ” ) ;
tu rn =1;
}

/∗ Pong p r o c e s s ∗/
while ( t r u e ){
while ( tu rn != 1 );
p r i n t ( ” pong ” ) ;
tu rn =0;
}
```

CODE-

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


int main(){
        int shmid;
        int key =123;
        int status;
        char *buffer;
        //creates shared memory segment
        if((shmid = shmget(key, 10, IPC_CREAT | 0666))==-1){
                printf("Cannot create shared memory\n");
                exit(0);
        }
        //attach it for use
        buffer = shmat(shmid, NULL, 0);

        int turn=0;   //initialise turn to 0 for pin process
sprintf(buffer, "%d", turn);

//loop to create two child processes
for(int i=0; i<2; i++){
        if(fork()!=0){
                //Parent process
        }
        else{
                //Child Process ping
                if(i==0){

                        while(1){
                                //printf("ping process\n");
                                sscanf(buffer, "%d", &turn);
                        while(turn!=0){
                                sscanf(buffer, "%d", &turn);
                        }
                        printf("Ping\n");
                        turn=1;
                        sprintf(buffer,"%d",turn);
                        sleep(1);
                 }

                }
                //Child process pong
                if(i==1){

                        while(1){
                                //printf("pong Process\n");
                        sscanf(buffer, "%d", &turn);
                        while(turn!=1){
                                sscanf(buffer, "%d", &turn);
                        }
                        printf("Pong\n");
                        turn=0;
                        sprintf(buffer,"%d",turn);
                        sleep(1);
                 }

                }

                exit(0);   //exit from child
        }
}
//two wait system calls for two child processes
wait(&status);
wait(&status);
shmdt(buffer);
return 0;
}
```

OUTPUT SCREEN-



```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$ ./1.out
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
ping
pong
^C
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$
```

QUESTION -2

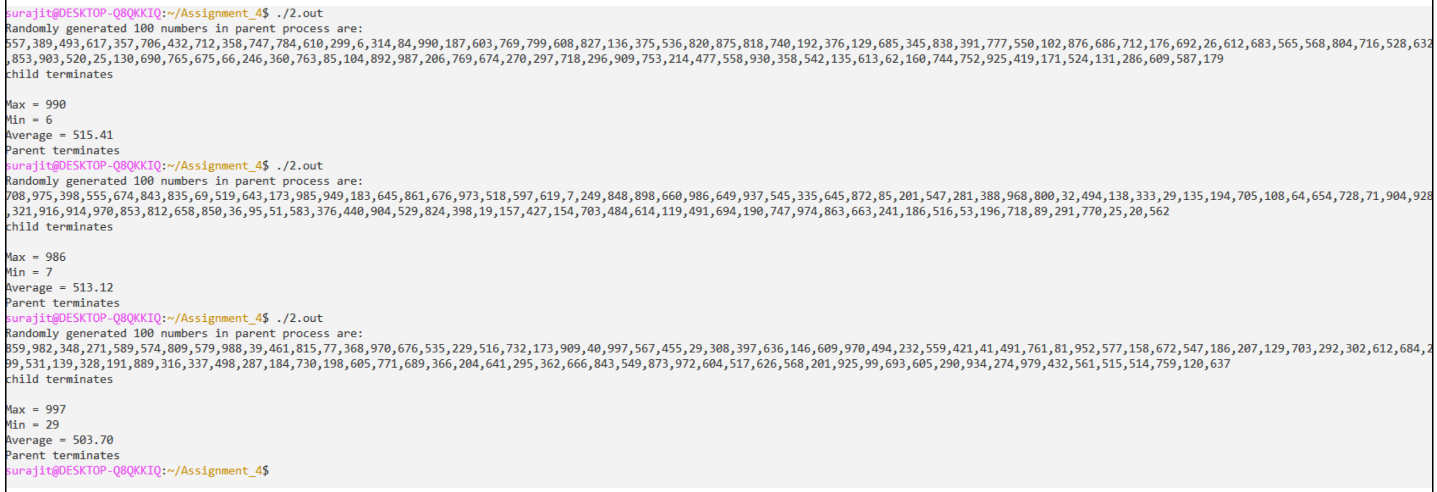
Write a C program which creates a child process. The parent and child process communicate using a shared memory segment. The parent process generates 100 random integers and writes it into the shared memory segment. The child process then computes the maximum, minimum and average of all these 100 numbers and writes the result back into the shared memory segment, from where the parent process reads the result and displays it. Add appropriate code to synchronize the parent and child process.

CODE-

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
#include<time.h>


int main(){
        int shmid;
        int key =220;
        int status,max,min,sum;
        float avg;
        char *buffer;
        //creates shared memory segment
        if((shmid = shmget(key, 420, IPC_CREAT | 0666))==-1){
                printf("Cannot create shared memory\n");
                exit(0);
        }
        //attach it for use
        buffer = shmat(shmid, NULL, 0);

        int turn=0;   //initialise turn to 0 for parent process
        sprintf(buffer, "%d", turn);

        if(fork()!=0){
               //parent process
               //waiting for child to give turn permission
               while(turn!=0){
                      sscanf(buffer, "%d" , &turn);
              }
              printf("Randomly generated 100 numbers in parent process are:\n");
              int i;
              //srand() must be used before rand() to produce differenct random numbers on
every execution
```

```
        //srand(val) start generating at different starting point and time(0) for current
time
        srand(time(0));
        //loop to generate 100 random numbers
        for(i=1;i<=100;i++)
        {
            int temp=rand()%1000;
            printf("%d",temp);
            //store numbers in shared memory
            sprintf(buffer+(4*i),"%d",temp);
            if(i!=100)
                printf(",");
            else
                printf("\n");
        }
        fflush(stdout);
        //release turn and pass the numbers to child process for operations
        turn=1;
        sprintf(buffer, "%d" ,turn);
        //waiting for child to give turn permission
        while(turn!=0){
            sscanf(buffer, "%d" , &turn);
        }
        sscanf(buffer+(4*101),"%d",&max);
        sscanf(buffer+(4*102),"%d",&min);
        sscanf(buffer+(4*103),"%f",&avg);
        sleep(1);
        printf("\nMax = %d\nMin = %d\nAverage = %.2f\n",max,min,avg);
        printf("Parent terminates\n");
    }
    else{
         //child process
         //waiting for parent to give turn permission
         while(turn!=1){
             sscanf(buffer, "%d" , &turn);
        }
        int val;
        sscanf(buffer+4, "%d" , &val);
        max=val;
        min=val;
        sum=val;
        int i;
        //loop to find min,max and  sum of values in shared memory
        for(i=2;i<=100;i++)
        {
            sscanf(buffer+(4*i),"%d",&val);
            if(val>max)
                max=val;
            if(val<min)
                min=val;
                sum+=val;
            }
            avg=(float)sum/100.0;
            //store max,min and average in shared memory
            sprintf(buffer+(4*101),"%d",max);
            sprintf(buffer+(4*102),"%d",min);
            sprintf(buffer+(4*103),"%.2f",avg);
            //release the turn and pass to parent process
            turn=0;
            sprintf(buffer, "%d" ,turn);
            printf("child terminates\n");
        }
        shmdt(buffer);
        return 0;
    }
```

OUTPUT SCREEN-



```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$ ./2.out
Randomly generated 100 numbers in parent process are:
557,389,493,617,357,706,432,712,358,747,784,610,299,6,314,84,990,187,603,769,799,608,827,136,375,536,820,875,818,740,192,376,129,685,345,838,391,777,550,102,876,686,712,176,692,26,612,683,565,568,804,716,528,632
,853,903,520,25,130,690,765,675,66,246,360,763,85,104,892,987,206,769,674,270,297,718,296,909,753,214,477,558,930,358,542,135,613,62,160,744,752,925,419,171,524,131,286,609,587,179
child terminates

Max = 990
Min = 6
Average = 515.41
Parent terminates
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$ ./2.out
Randomly generated 100 numbers in parent process are:
708,975,398,555,674,843,835,69,519,643,173,985,949,183,645,861,676,973,518,597,619,7,249,848,898,660,986,649,937,545,335,645,872,85,201,547,281,388,968,800,32,494,138,333,29,135,194,705,108,64,654,728,71,904,928
,321,916,914,970,853,812,658,850,36,95,51,583,376,440,904,529,824,398,19,157,427,154,703,484,614,119,491,694,190,747,974,863,663,241,186,516,53,196,718,89,291,770,25,20,562
child terminates

Max = 986
Min = 7
Average = 513.12
Parent terminates
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$ ./2.out
Randomly generated 100 numbers in parent process are:
859,982,348,271,589,574,809,579,988,39,461,815,77,368,970,676,535,229,516,732,173,909,40,997,567,455,29,308,397,636,146,609,970,494,232,559,421,41,491,761,81,952,577,158,672,547,186,207,129,703,292,302,612,684,2
99,531,139,328,191,889,316,337,498,287,184,730,198,605,771,689,366,204,641,295,362,666,843,549,873,972,604,517,626,568,201,925,99,693,605,290,934,274,979,432,561,515,514,759,120,637
child terminates

Max = 997
Min = 29
Average = 503.70
Parent terminates
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$
```

QUESTION -3

Implement the solution to the producer-consumer problem using shared variables.

CODE-

PRODUCER CODE

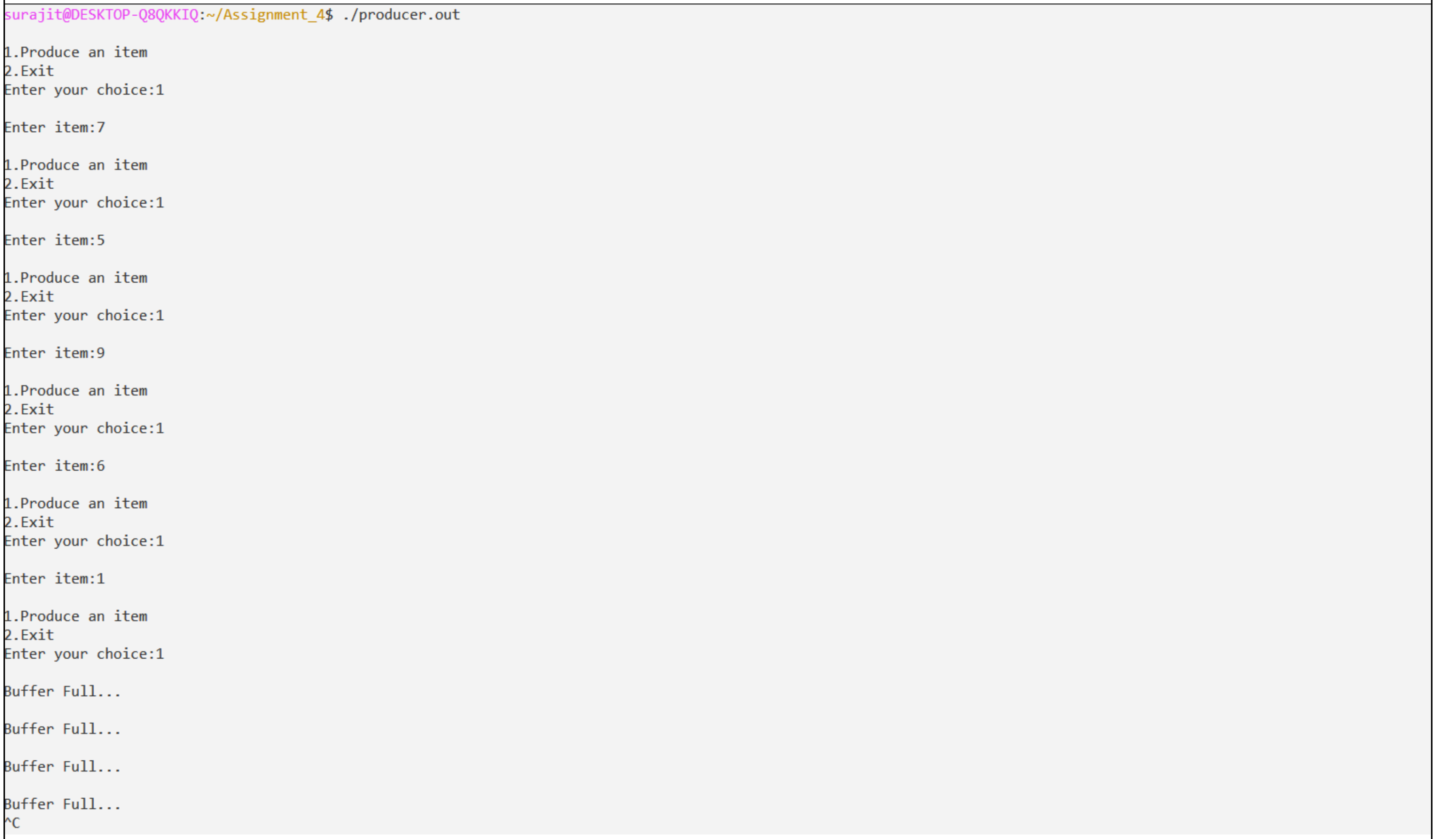
```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
#include<time.h>


int main(){
        int shmid,key=2022,ch,item;
        char *buffer;

        //creates shared memory segment of 20 bytes
        //full empty 5 elements
        if((shmid = shmget(key, 28 , IPC_CREAT | 0666))==-1){
                printf("Cannot create shared memory\n");
                exit(0);
        }
        //attach it for use
        buffer = shmat(shmid, NULL, 0);


        int full,empty;
        full=0;
        empty=5;
        sprintf(buffer,"%d",full);
        sprintf(buffer+4,"%d",empty);
        //do-while loop to produce items repetatively
        do{
              printf("\n1.Produce an item\n2.Exit\nEnter your choice:");
              scanf("%d",&ch);
              switch(ch){
                    case 1:
                          sscanf(buffer,"%d",&full);
                          sscanf(buffer+4,"%d",&empty);
                          while(empty==0){
                                printf("\nBuffer Full...\n");
                                sleep(1);
                                sscanf(buffer+4,"%d",&empty);
                          }
                          printf("\nEnter item:");
                          scanf("%d",&item);
                          full+=1;
                          empty-=1;
                    sprintf(buffer+4+full*4,"%d",item);
                    sprintf(buffer,"%d",full);
                    sprintf(buffer+4,"%d",empty);
                    break;
              case 2:
                   printf("\nProgram Exits...\n");
                   break;
              default:
                   printf("\nWrong Choice...");
         }
    }while(ch!=2);
    printf("\nItems produces:%d, Buffer slots remains:%d\n",full,empty);
    return 0;
     }
```

CONSUMER CODE

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
#include<time.h>


int main(){
     int shmid,key=2022,sz,ch,item;
     char *buffer;
     //creates shared memory segment
     if((shmid = shmget(key, 28 , IPC_CREAT | 0666))==-1){
             printf("Cannot create shared memory\n");
             exit(0);
     }
     //attach it for use
     buffer = shmat(shmid, NULL, 0);


     int full,empty;

     //do-while loop to produce items repetatively
     do{
          printf("\n1.Consume an item\n2.Exit\nEnter your choice:");
          scanf("%d",&ch);
          switch(ch){
               case 1:
                    sscanf(buffer,"%d",&full);
                    sscanf(buffer+4,"%d",&empty);
                    while(full==0){
                         printf("\nBuffer Empty...\n");
                         sleep(1);
                         sscanf(buffer,"%d",&full);
                    }

                    sscanf(buffer+4+full*4,"%d",&item);
                    printf("\nItem Consumed:%d\n",item);
```

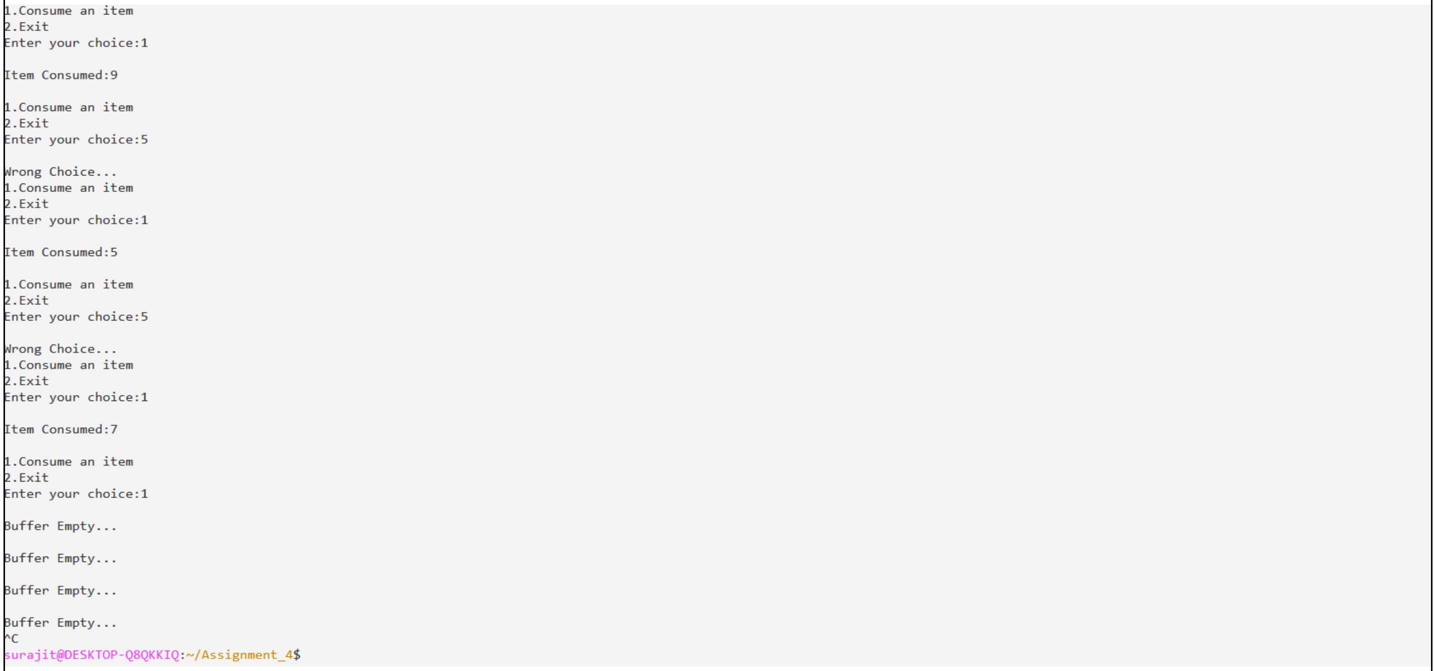
```
                        full-=1;
                        empty+=1;
                        sprintf(buffer,"%d",full);
                        sprintf(buffer+4,"%d",empty);
                        break;
                case 2:
                       printf("\nProgram Exits...\n");
                       break;
                default:
                       printf("\nWrong Choice...");
          }
     }while(ch!=2);
     printf("\nItems produces:%d, Buffer slots remains:%d\n",full,empty);
     return 0;
}
```

OUTPUT SCREEN-



```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$ ./producer.out

1.Produce an item
2.Exit
Enter your choice:1

Enter item:7

1.Produce an item
2.Exit
Enter your choice:1

Enter item:5

1.Produce an item
2.Exit
Enter your choice:1

Enter item:9

1.Produce an item
2.Exit
Enter your choice:1

Enter item:6

1.Produce an item
2.Exit
Enter your choice:1

Enter item:1

1.Produce an item
2.Exit
Enter your choice:1

Buffer Full...

Buffer Full...

Buffer Full...

Buffer Full...
^C
```

```
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$ ./consumer.out

1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:1

1.Consume an item
2.Exit
Enter your choice:6

Wrong Choice...
1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:6

1.Consume an item
2.Exit
Enter your choice:9

Wrong Choice...
1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:9

1.Consume an item
2.Exit
Enter your choice:5

Wrong Choice...
1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:5

1.Consume an item
2.Exit
Enter your choice:5

Wrong Choice...
1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:7
```



```
1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:9

1.Consume an item
2.Exit
Enter your choice:5

Wrong Choice...
1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:5

1.Consume an item
2.Exit
Enter your choice:5

Wrong Choice...
1.Consume an item
2.Exit
Enter your choice:1

Item Consumed:7

1.Consume an item
2.Exit
Enter your choice:1

Buffer Empty...

Buffer Empty...

Buffer Empty...

Buffer Empty...
^C
surajit@DESKTOP-Q8QKKIQ:~/Assignment_4$
```