Name : Rajvardhan Reddy
Reg No : 180905093
Sec : B
Roll No : 19

OS LAB – 6 : IPC – 2: Message Queue, Shared Memory

Lab Exercises:

P1) Process A wants to send a number to Process B. Once received, Process B has to check whether the number is palindrome or not. Write a C program to implement this interprocess communication using a message queue.

Code :

Sender :

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#include <errno.h>

#define SIZE 128

typedef struct message {
   long mtype;
   char c[SIZE];
}MESSAGE;

int main()
{
        int qt;
        MESSAGE msg;
        if((qt = msgget(1272, 0)) < 0)
        {
                perror("Error in msgget()");
                exit(EXIT_FAILURE);
        }
        printf("Enter the number: ");
        fgets(msg.c, SIZE, stdin);
        msg.mtype = 1;
```

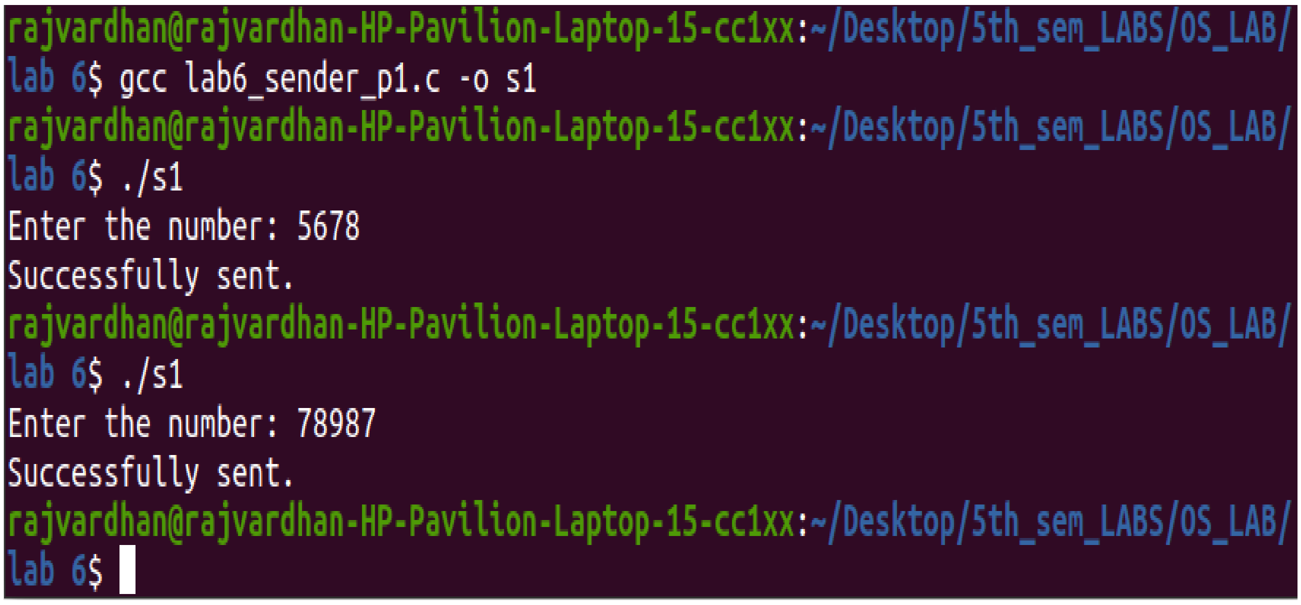
```
	if(msgsnd(qt, &msg, sizeof(MESSAGE), 0) < 0)
	{
		perror("Error in msgsnd()");
		exit(1);
		}
	printf("Successfully sent.\n");
	return 0;
}
```

Receiver :

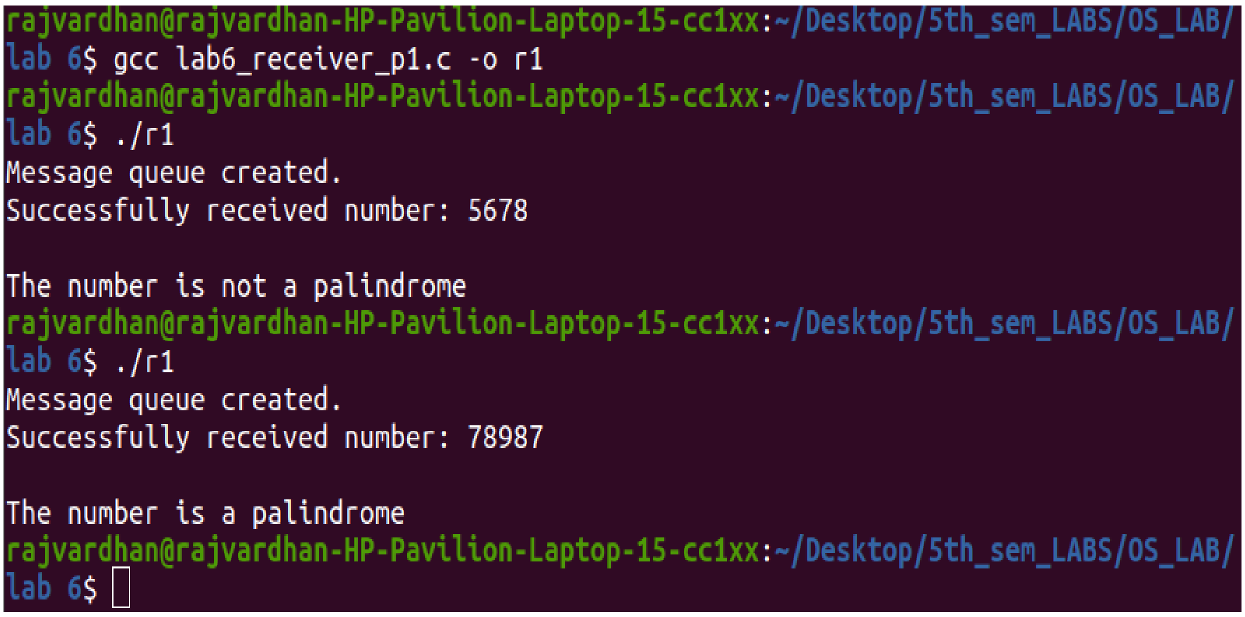
```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/msg.h>
#include <sys/ipc.h>
#include <errno.h>

#define SIZE 128

typedef struct message {
	long mtype;
	char c[SIZE];
}MESSAGE;

int isPalindrome(int n)
{
	int r = 0, num = n;
	while (n > 0)
	{
		r = r * 10 + n % 10;
		n = n / 10;
	}
	if (r == num)
		return 1;
	else
		return 0;
}

int main()
{
	int qt;
	MESSAGE msg;
	if ((qt = msgget(1272, IPC_CREAT | IPC_EXCL | 0600)) < 0)
	{
```

```
                perror("Error in msgget()");
                exit(EXIT_FAILURE);
        }
        printf("Message queue created.\n");
        if (msgrcv(qt, &msg, sizeof(MESSAGE), 0, 0) < 0)
        {
                perror("Error in msgrcv()");
                exit(EXIT_FAILURE);
        }
        printf("Successfully received number: %s\n", msg.c);

        int num = atoi(msg.c);
        if (isPalindrome(num))
                printf("The number is a palindrome\n");
        else
                printf("The number is not a palindrome\n");

        if(msgctl(qt, IPC_RMID, NULL)==-1)
        {
                fprintf(stderr, "IPC_RMID failed\n");
                exit(EXIT_FAILURE);
        }
        exit(EXIT_SUCCESS);
}
```

Output :



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ gcc lab6_sender_p1.c -o s1
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./s1
Enter the number: 5678
Successfully sent.
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./s1
Enter the number: 78987
Successfully sent.
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$
```



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ gcc lab6_receiver_p1.c -o r1
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./r1
Message queue created.
Successfully received number: 5678

The number is not a palindrome
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./r1
Message queue created.
Successfully received number: 78987

The number is a palindrome
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ 
```

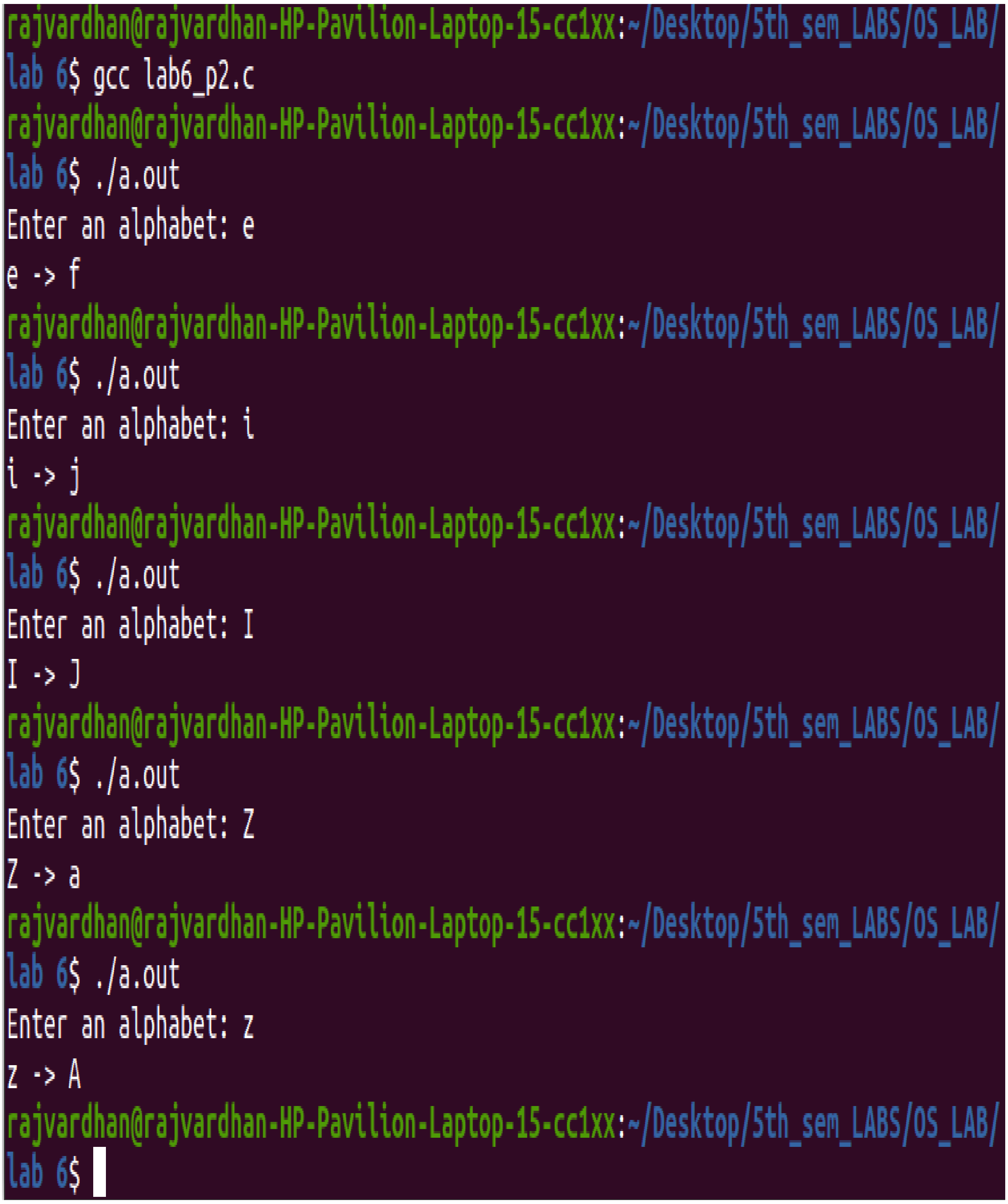
P2) Implement a parent process, which sends an English alphabet to a child process using shared memory. The child process responds with the next English alphabet to the parent. The parent displays the reply from the Child.

<u>Code :</u>

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

#define SIZE 2

void getNextAlphabet (char *a)
{
        char *n = (char *)calloc(2, sizeof(char));
        n[1] = '\0';
        if (*a == 'Z')
                n[0] = 'a';
        else if (*a == 'z')
                n[0] = 'A';
        else
                n[0] = *a + 1;
        *a = *n;
```

```
}

int main (int argc, const char * argv [])
{
        int shmid = shmget(IPC_PRIVATE, SIZE, S_IRUSR | S_IWUSR);
        if(shmid==-1)
        {
                fprintf(stderr, "shmget failed\n");
                exit(EXIT_FAILURE);
        }

        char *shared_memory = (char *)shmat(shmid, NULL, 0);
        if(shared_memory==(void*)-1)
        {
                fprintf(stderr, "shmat failed\n");
                exit(EXIT_FAILURE);
        }
        *shared_memory = '\0';

        printf("Enter an alphabet: ");
        scanf("%c", shared_memory);

        pid_t pid = fork();
        if (pid == 0) //child process
        {
                while (*shared_memory == '\0');
                getNextAlphabet(shared_memory);
                exit(0);
        }
        else //parent process
        {
                printf("%s -> ", shared_memory);
                wait(NULL);
                printf("%s\n", shared_memory);
        }

        if(shmdt(shared_memory)==-1)
        {
                fprintf(stderr, "shmdt failed\n");
                exit(EXIT_FAILURE);
        }
        if(shmctl(shmid, IPC_RMID, NULL)==-1)
        {
                fprintf(stderr, "IPC_RMID failed\n");
                exit(EXIT_FAILURE);
        }
        exit(EXIT_SUCCESS);
}
```

Output :



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ gcc lab6_p2.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./a.out
Enter an alphabet: e
e -> f
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./a.out
Enter an alphabet: i
i -> j
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./a.out
Enter an alphabet: I
I -> J
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./a.out
Enter an alphabet: Z
Z -> a
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./a.out
Enter an alphabet: z
z -> A
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ 
```

Additional Exercises:

P3) Write a producer-consumer program in C in which producer writes a set of words into shared memory and then consumer reads the set of words from the shared memory. The shared memory need to be detached and deleted after use.

Code :

shm_com.h :

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define TEXT_SZ 2048

struct shared_use_st{
	int written_by_you;
	char some_text[TEXT_SZ];
};
```

Producer :

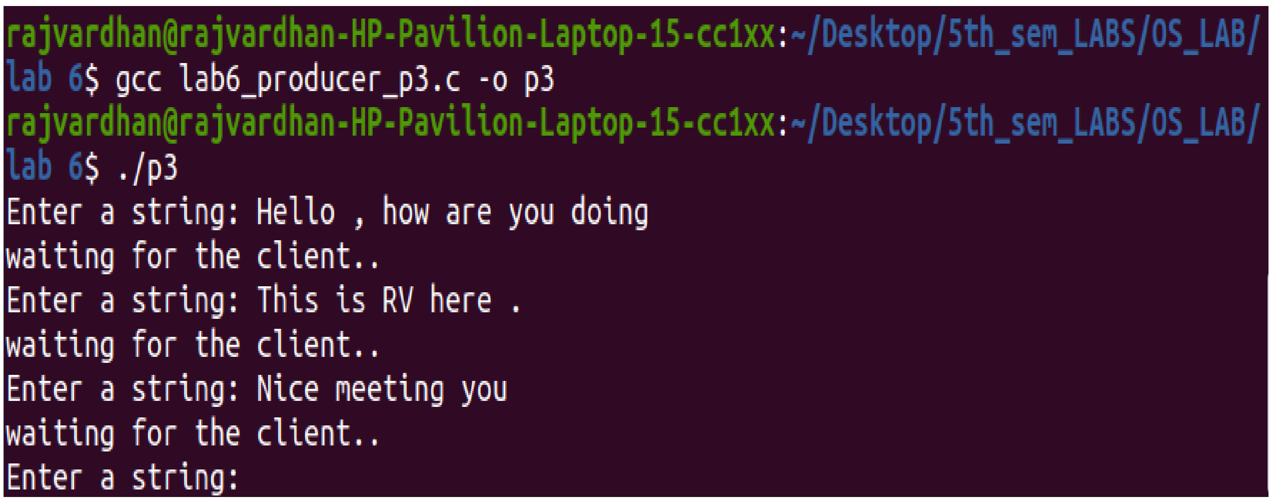
```
#include "shm_com.h"
int main(){
	int running = 1;
	void *shared_memory = (void*)0;
	struct shared_use_st *shared_stuff;
	char buffer[BUFSIZ];
	int shmid;
	shmid = shmget((key_t)1235,sizeof(struct shared_use_st),0666|
IPC_CREAT);
	if(shmid==-1){
		fprintf(stderr,"shmget failed\n");
		exit(EXIT_FAILURE);
	}
	shared_memory = shmat(shmid,(void*)0,0);
	if(shared_memory==(void*)-1){
		fprintf(stderr,"shmat failed\n");
		exit(EXIT_FAILURE);
	}
	shared_stuff = (struct shared_use_st*)shared_memory;
	while(running){
		while(shared_stuff->written_by_you == 1){
			sleep(1);
			printf("waiting for the client..\n");
			}
		printf("Enter a  string:");
		fgets(buffer,BUFSIZ,stdin);
		strncpy(shared_stuff->some_text,buffer,TEXT_SZ);
		shared_stuff->written_by_you = 1;
		if(strncmp(buffer,"end",3)==0)
			running = 0;

	}
	if(shmdt(shared_memory)==-1){
		fprintf(stderr,"shmdt failed\n");
		exit(EXIT_FAILURE);
	}
	exit(EXIT_SUCCESS);
}
```

Consumer :

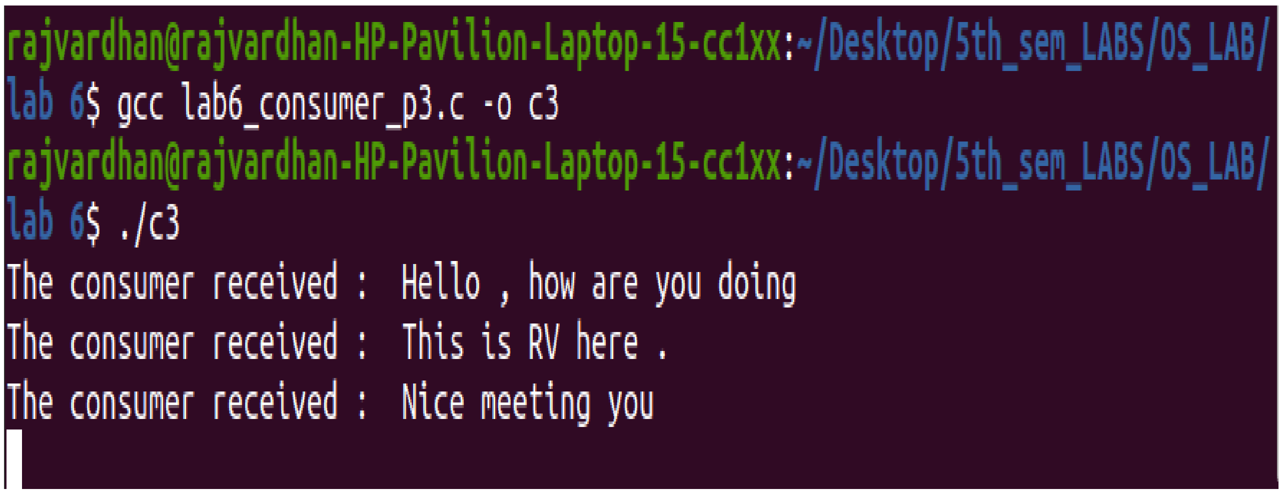
```
#include "shm_com.h"
int main(){
   int running = 1;
   void *shared_memory = (void*)0;
   struct shared_use_st *shared_stuff;
   int shmid;
   srand((unsigned int)getpid());
   shmid = shmget((key_t)1235,sizeof(struct shared_use_st),0666|
IPC_CREAT);
   if(shmid==-1){
      fprintf(stderr,"shmget failed\n");
      exit(EXIT_FAILURE);
   }
   shared_memory = shmat(shmid,(void*)0,0);
```

```
    if(shared_memory == (void*)-1){
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    //printf("Memory attached at %X\n",(int)shared_memory);
    shared_stuff = (struct shared_use_st*)shared_memory;
    shared_stuff->written_by_you = 0;
    while(running){
        if(shared_stuff->written_by_you){
            printf("The consumer received : %s",shared_stuff-
>some_text);
            sleep(rand()%4);
            shared_stuff->written_by_you = 0;
            if(strncmp(shared_stuff->some_text,"end",3)==0){
                running = 0;
            }
        }
    }
    if(shmdt(shared_memory)==-1){
        fprintf(stderr,"shmdt failed\n");
        exit(EXIT_FAILURE);
    }
    if(shmctl(shmid,IPC_RMID,0)==-1){
        fprintf(stderr,"shmctl(IPC_RMID) failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);

}
```

<u>Output :</u>



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ gcc lab6_producer_p3.c -o p3
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./p3
Enter a string: Hello , how are you doing
waiting for the client..
Enter a string: This is RV here .
waiting for the client..
Enter a string: Nice meeting you
waiting for the client..
Enter a string:
```



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ gcc lab6_consumer_p3.c -o c3
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./c3
The consumer received :  Hello , how are you doing
The consumer received :  This is RV here .
The consumer received :  Nice meeting you
```

P4) Write a program which creates a message queue and writes message into queue which contains number of users working on the machine along with observed time in hours and minutes. This is repeated for every 10 minutes. Write another program which reads this information form the queue and calculates on average in each hour how many users are working.

Code :

Process 1 :

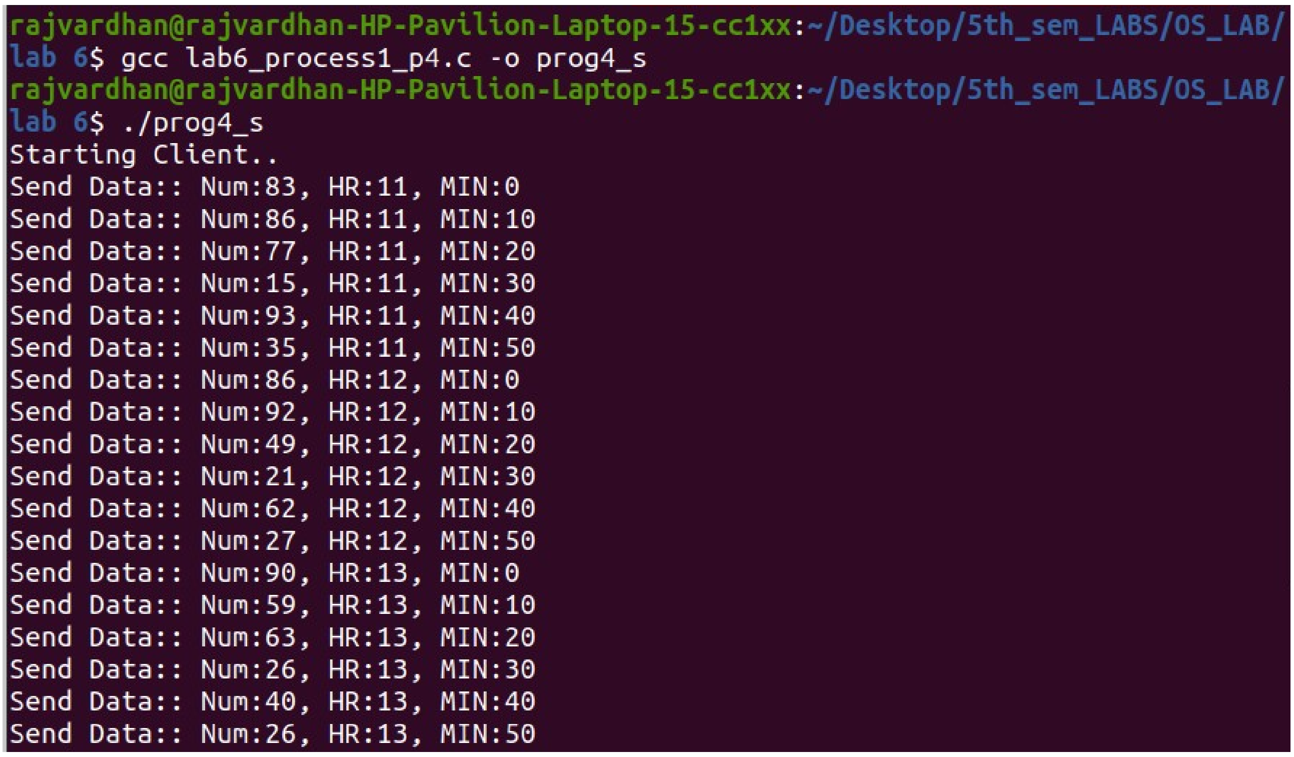
```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>

#include <sys/msg.h>
#define MSG_TO_RECV 0
#define MSG_KEY 1234

typedef struct msg_st
{
long int msg_type;
int num;
int hr;
int min;
} MSG;
```

```
int main()
{
printf("Starting Client..\n");
int active = 1;
MSG msgSent;
char buff[BUFSIZ];
int msgid = msgget((key_t)MSG_KEY, 0666 | IPC_CREAT);

if (msgid == -1)
{
	perror("msgget()");
	exit(EXIT_FAILURE);
}

int hr = 11;
int min = 0;

while (active)
{
int num = rand() % 100;
msgSent.msg_type = 1;
msgSent.num = num;
msgSent.hr = hr;
msgSent.min = min;
printf("Send Data:: Num:%d, HR:%d, MIN:%d\n",num, hr, min);

if (msgsnd(msgid, (void *)&msgSent, sizeof(int) * 3, 0) == -1)
{
	perror("msgsnd()");
	exit(EXIT_FAILURE);
}
if (msgSent.num == -1)
{
	active = 0;
	printf("Shutting down client...\n");
	break;
}
min += 10;
if (min >= 60)
{
	min = 0;
	hr++;
}
sleep(1); //assume 1 sec = 10 min
}
exit(EXIT_SUCCESS);
}
```

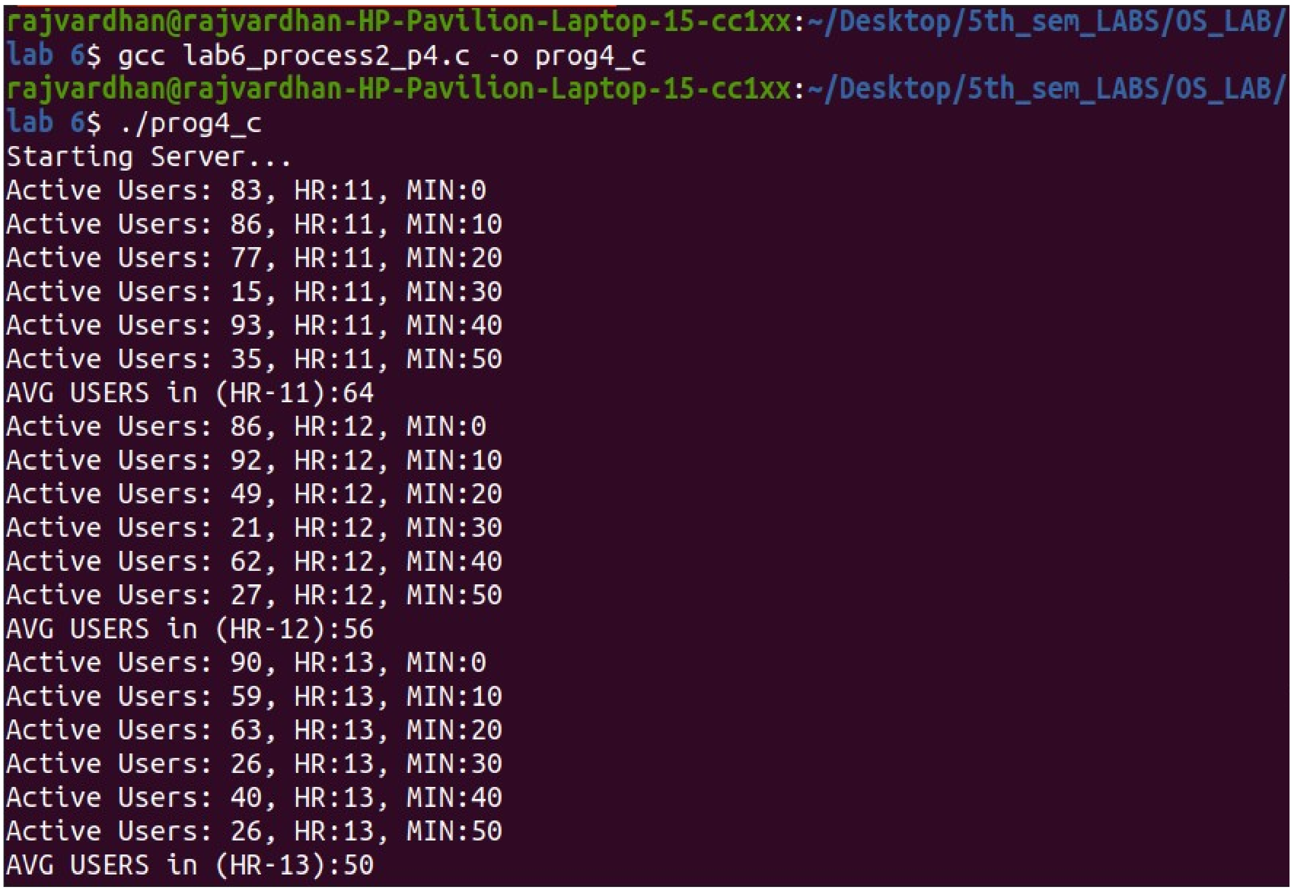
Process 2 :

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MSG_TO_RECV 0
#define MSG_KEY 1234

typedef struct msg_st
{
long int msg_type;
int num;
int hr;
int min;
} MSG;

int main()
{
int active = 1;
MSG msgRecv;
char buff[BUFSIZ];
int msgid = msgget((key_t)MSG_KEY, 0666 | IPC_CREAT);

if (msgid == -1)
{
        perror("msgget()");
        exit(EXIT_FAILURE);
}

printf("Starting Server...\n");
int count = 0;
int sum = 0;

while (active)
{
        if (msgrcv(msgid, (void *)&msgRecv, BUFSIZ,
        MSG_TO_RECV, 0) == -1)
{
perror("msgrecv()");

exit(EXIT_FAILURE);

}
if (msgRecv.num == -1)
{
        active = 0;
        printf("Shutting down Server...\n");
        break;
}
printf("Active Users: %d, HR:%d, MIN:%d\n", msgRecv.num,
msgRecv.hr, msgRecv.min);

count++;
sum += msgRecv.num;

if (count == 6)
{
        printf("AVG USERS in (HR-%d):%d\n",
        msgRecv.hr, sum / count);
        count = 0;
        sum = 0;
}

}

if (msgctl(msgid, IPC_RMID, 0) == -1)
{
        perror("msgctl");
        exit(1);
}

exit(EXIT_SUCCESS);

}
```

Output :



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ gcc lab6_process1_p4.c -o prog4_s
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./prog4_s
Starting Client..
Send Data:: Num:83, HR:11, MIN:0
Send Data:: Num:86, HR:11, MIN:10
Send Data:: Num:77, HR:11, MIN:20
Send Data:: Num:15, HR:11, MIN:30
Send Data:: Num:93, HR:11, MIN:40
Send Data:: Num:35, HR:11, MIN:50
Send Data:: Num:86, HR:12, MIN:0
Send Data:: Num:92, HR:12, MIN:10
Send Data:: Num:49, HR:12, MIN:20
Send Data:: Num:21, HR:12, MIN:30
Send Data:: Num:62, HR:12, MIN:40
Send Data:: Num:27, HR:12, MIN:50
Send Data:: Num:90, HR:13, MIN:0
Send Data:: Num:59, HR:13, MIN:10
Send Data:: Num:63, HR:13, MIN:20
Send Data:: Num:26, HR:13, MIN:30
Send Data:: Num:40, HR:13, MIN:40
Send Data:: Num:26, HR:13, MIN:50
```



```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ gcc lab6_process2_p4.c -o prog4_c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LAB/
lab 6$ ./prog4_c
Starting Server...
Active Users: 83, HR:11, MIN:0
Active Users: 86, HR:11, MIN:10
Active Users: 77, HR:11, MIN:20
Active Users: 15, HR:11, MIN:30
Active Users: 93, HR:11, MIN:40
Active Users: 35, HR:11, MIN:50
AVG USERS in (HR-11):64
Active Users: 86, HR:12, MIN:0
Active Users: 92, HR:12, MIN:10
Active Users: 49, HR:12, MIN:20
Active Users: 21, HR:12, MIN:30
Active Users: 62, HR:12, MIN:40
Active Users: 27, HR:12, MIN:50
AVG USERS in (HR-12):56
Active Users: 90, HR:13, MIN:0
Active Users: 59, HR:13, MIN:10
Active Users: 63, HR:13, MIN:20
Active Users: 26, HR:13, MIN:30
Active Users: 40, HR:13, MIN:40
Active Users: 26, HR:13, MIN:50
AVG USERS in (HR-13):50
```