**OS Lab**

**Session 3 – Lab 6 (15/12/2020)**

Parthivi Choubey CSE – B - 5th semester

180905456 Roll. no. - 60

**Question 1**

**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");

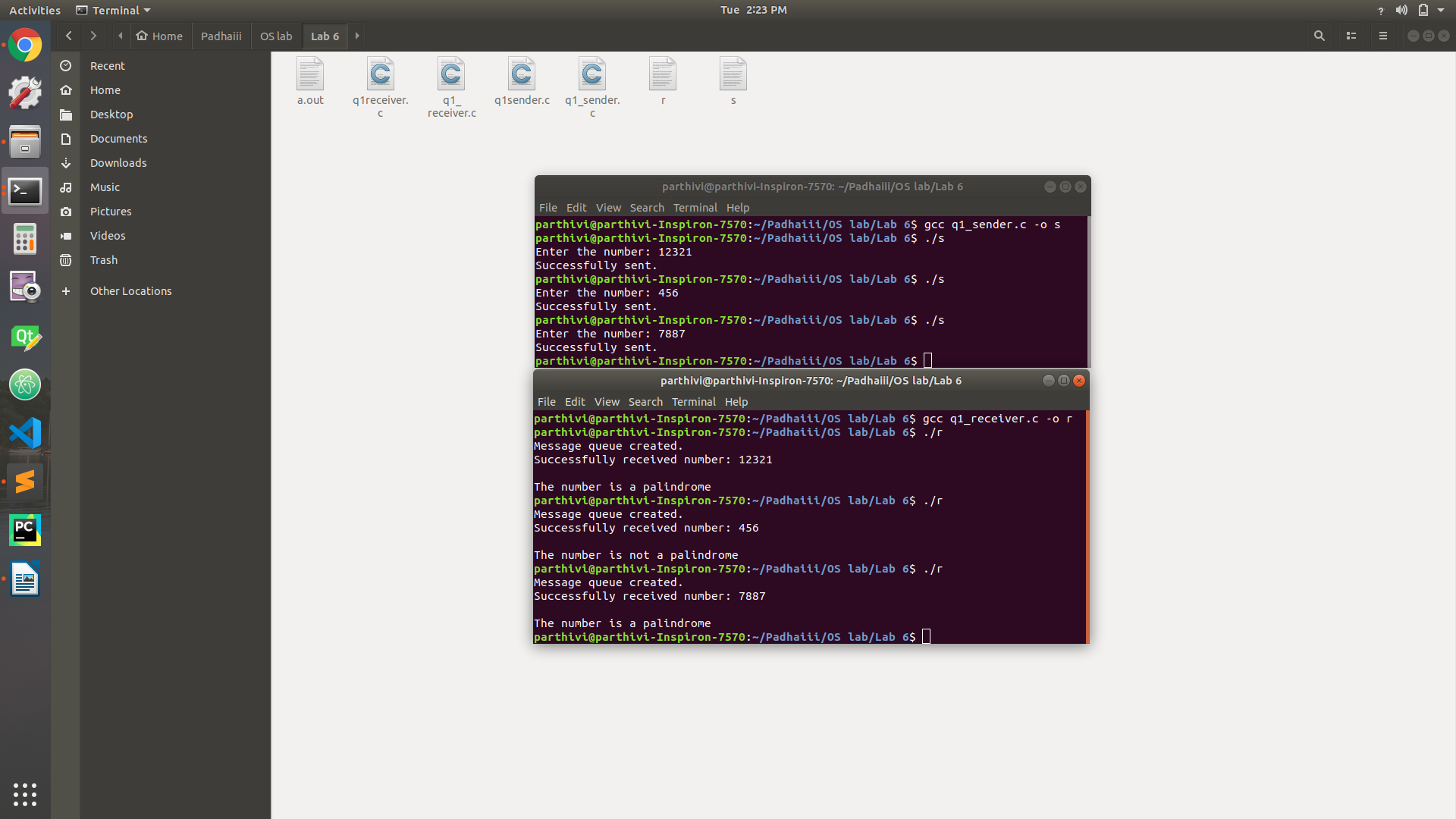
exit(EXIT\_FAILURE);

}

exit(EXIT\_SUCCESS);

}

**Output**



**Question 2**

**Code**

#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**

