**OPERATING SYSTEM WEEK 6**

**Q1.**

PROGRAM:

**Process A:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<errno.h>

#include<unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#define MAX\_TEXT 512

struct my\_msg{

long int my\_msg\_type;

int num;

};

int main(int argc, char const \*argv[])

{

int running = 1;

struct my\_msg some\_data;

int msgid;

char buffer[BUFSIZ];

msgid = msgget((key\_t) 1234, 0666 | IPC\_CREAT);

if(msgid == -1){

fprintf(stderr, "mssget failed with error: %d\n", errno);

exit(EXIT\_FAILURE);

}

int num;

while(running){

printf("Enter an integer(-1 to exit):");

scanf("%d", &num);

some\_data.my\_msg\_type = 1;

some\_data.num = num;

if(msgsnd(msgid, (void\*)& some\_data, sizeof(int), 0) == -1){

fprintf(stderr, "msgsnd failed with error:%d\n", errno);

exit(EXIT\_FAILURE);

}

if(num == -1)

running = 0;

}

exit(EXIT\_SUCCESS);

return 0;

}

**Process B:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<errno.h>

#include<unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include<stdbool.h>

struct my\_msg{

long int my\_msg\_type;

int num;

};

void isPalindrome(int n){

int reversedN = 0, remainder;

int originalN = n;

while (n != 0) {

remainder = n % 10;

reversedN = reversedN \* 10 + remainder;

n /= 10;

}

if (originalN == reversedN)

printf("Palindrome!\n");

else

printf("Not a Palindrome!\n");

return;

}

int main(int argc, char const \*argv[])

{

int running = 1;

int msgid;

struct my\_msg some\_data;

long int msgToRecieve = 0;

msgid = msgget((key\_t)1234, 0666 | IPC\_CREAT);

if(msgid == -1){

fprintf(stderr, "mssget failed with error: %d\n", errno);

exit(EXIT\_FAILURE);

}

int num;

while(running){

if(msgrcv(msgid, (void\*)& some\_data, sizeof(int), msgToRecieve, 0) == -1){

fprintf(stderr, "msgrcv failed with error:%d\n", errno);

exit(EXIT\_FAILURE);

}

// printf("You wrote: %s", some\_data.buffer);

num = some\_data.num;

if(num == -1){

running = 0;

printf("Exiting!\n");

}

else

isPalindrome(num);

}

if(msgctl(msgid, IPC\_RMID, 0) == -1){

fprintf(stderr, "msgctl(IPC\_RMID) failed\n");

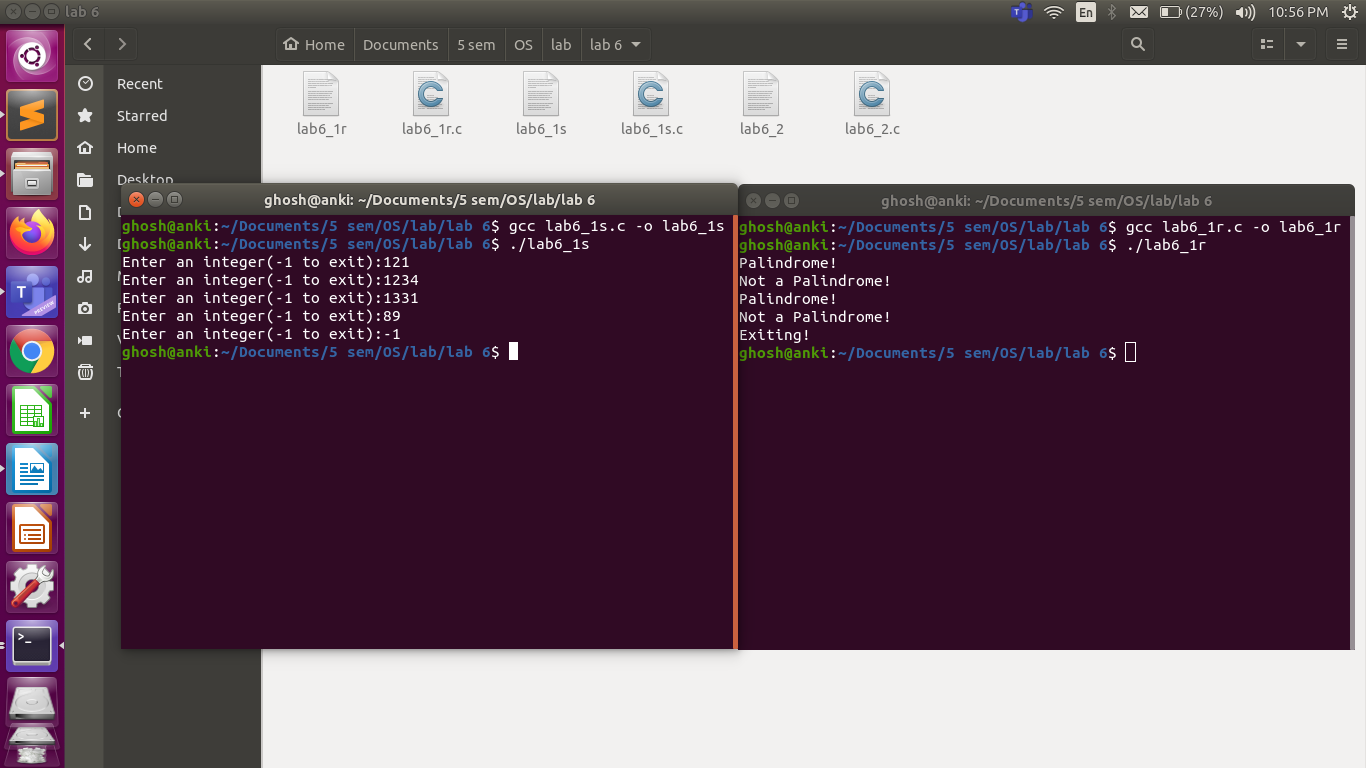
exit(EXIT\_FAILURE);

}

exit(EXIT\_SUCCESS);

return 0;

}  
  
OUTPUT:



**Q2.**

PROGRAM:

#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/shm.h>

struct shared\_use

{

char c;

char next\_char;

};

int main()

{

int running = 1;

pid\_t pid;

void \*shared\_memory = (void \*)0;

int shmid;

struct shared\_use \*st;

srand((unsigned int)getpid());

shmid = shmget((key\_t)1234,sizeof(struct shared\_use),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);

}

st = (struct shared\_use \*)shared\_memory;

char ch;

printf("Enter a character: \n");

scanf("%c",&ch);

st->c=ch;

printf("You entered: %c\n",st->c);

pid = fork();

if(pid == -1)

{

fprintf(stderr, "%d\n",errno);

exit(1);

}

else if(pid == 0)

{

//Child process

printf("Current character: %c\n",st->c);

st->c++;

}

else

{

//Parent process

printf("\nWaiting for Child process to increment the character\n");

sleep(2);

printf("Updated character: %c\n",st->c);

}

if(shmdt(shared\_memory)==-1)

{

fprintf(stderr, "shmdt failed\n");

exit(EXIT\_FAILURE);

}

}  
  
OUTPUT:

