PART A

1)BRANCH :CREATE TABLE BRANCH (Branchid integer primary key, Branchname varchar2 (15) not null, HOD varchar2 (10));
STUDENT :CREATE TABLE STUDENT (USN varchar2 (15) primary key, Name varchar2 (15) not null, Address varchar2 (15) not null, Branchid integer references branch, SEM varchar2 (10));
BOOK: CREATE TABLE BOOK (Bookid varchar2 (10) primary key, Bookname varchar2 (15) not null, Authorid varchar2 (10) reference author, publisher varchar2 (20) not null, Branchid number references branch);
AUTHOR: CREATE TABLE AUTHOR (Authorid varchar2 (10) primary key, Authername varchar2 (15) not null, country varchar2 (15), Age integer);
BORROW:CREATE TABLE BORROW (USN Varchar2 (15) references student, Bookid varchar2 (10) references book, Borrowed_Date date, primary key (USN, Bookid));

2) a. details of students who are all studying in 2nd sem MCA : SELECT * FROM STUDENT S, BRANCH B WHERE S.BRANCHID=B.BRANCHID AND S.SEM='II SEM' AND B.BRANCHNAME='MCA';
b.studens who not borrowed any books: SELECT * FROM STUDENT S WHERE S.USN NOT IN(SELECT B.USN FROM BORROW B);

3) a. Display the USN, Student_name, Branch_name, Book_name, Author_name, Books_Borrowed_Date of 2nd sem MCA students who borrowed books : SELECT S.USN, S.NAME, S.SEM, BR.BRANCHNAME, BK.BOOKNAME, A.AUTHERNAME, B.BORROWED_DATE FROM STUDENT S, BRANCH BR, BOOK BK, AUTHOR A, BORROW B WHERE S.BRANCHID=BR.BRANCHID AND A.AUTHORID=BK.AUTHORID AND B.USN=S.USN AND BK.BOOKID=B.BOOKID AND S.SEM='II SEM' AND BR.BRANCHNAME='MCA';
b.student details who borrowed books of more than 1 Author: SELECT A.AUTHERNAME, COUNT(DISTINCT BK.BOOKID) AS "NO OF BOOKS" FROM AUTHOR A, BOOK BK WHERE A.AUTHORID=BK.AUTHORID GROUP BY A.AUTHERNAME;

4) a.student details who borrowed more than 2 books : SELECT S.NAME FROM STUDENT S, BORROW B WHERE S.USN=B.USN GROUP BY S.NAME HAVING COUNT (DISTINCT B.BOOKID) > 2;
b. student details who borrowed books of more than 1 Author: SELECT S.NAME, COUNT (DISTINCT BK.BOOKID) FROM STUDENT S, BOOK BK, BORROW B WHERE S.USN=B.USN AND B.BOOKID = BK.BOOKID GROUP BY S.NAME HAVING COUNT (DISTINCT BK.AUTHORID) > 1;

5) a. book names in descending order of their names: SELECT * FROM BOOK ORDER BY BOOKNAME DESC;
b. details of students who borrowed books which r all published by same publisher: SELECT S.NAME, COUNT (BK.PUBLISHER) FROM STUDENT S, BOOK BK, BORROW B WHERE S.USN=B.USN AND B.BOOKID=BK.BOOKID GROUP BY S.NAME;

6) a.Creating Tables (with Constraints): CREATE TABLE STUDENT2 (USN VARCHAR(10) PRIMARY KEY, NAME VARCHAR(20) NOT NULL, DOB DATE, BRANCH VARCHAR(10) NOT NULL, MARK1 INTEGER(3) NOT NULL, MARK2 INTEGER(3) NOT NULL, MARK3 INTEGER(3) NOT NULL, TOTAL INTEGER(4), GPA DECIMAL(4,2));
b. Inserting the value in the record :INSERT INTO STUDENT2 VALUES('SCA202201','SANJANA','2004-08-24','BCA',85,96,97,NULL,NULL);
c. Display the entered value in student2 table: SELECT * FROM STUDENT2;
d. Updating the record (for calculate Total) : UPDATE STUDENT2 SET TOTAL = MARK1 + MARK2 + MARK3;

7) a. Find the GPA score of all the students: UPDATE STUDENT2 SET GPA = (TOTAL*100)/300;
b. Find students who born on a particular year of birth from the date of birth column : SELECT USN, NAME, BRANCH, DOB FROM STUDENT2 WHERE DOB LIKE '2004%';

8)a. students who are studying in a particular branch of study:SELECT USN,NAME, BRANCH, DOB FROM STUDENT2 WHERE BRANCH='BCA';
b. Find max GPA score of the student branch-wise : SELECT BRANCH, MAX (GPA) FROM STUDENT2 GROUP BY BRANCH;

9) a.students whose name starts with the alphabet "S" : SELECT * FROM STUDENT2 WHERE NAME LIKE 'S%';
b. Update column total by adding columns mark1, mark2, mark3:UPDATE STUDENT2 SET TOTAL = MARK1 + MARK2 + MARK3;

10)a. students whose name ends with alphabets "AR": SELECT NAME FROM STUDENT2 WHERE NAME LIKE '%AR";
b. Delete the student details whose USN is greater than 1001: SELECT * FROM STUDENT WHERE USN > '1001';

PART B

1)EMPLOYEE:CREATE TABLE EMPLOYEE (FNAME VARCHAR (15) NOT NULL, MINT CHAR, LNAME VARCHAR (15) NOT NULL, SSN CHAR (9) NOT NULL, BDATE DATE, ADDRESS VARCHAR (30), SEX CHAR, SALARY DECIMAL (10, 2), SUPER_SSN CHAR (9), DNO INTEGER NOT NULL, PRIMARY KEY (SSN), FOREIGN KEY (SUPER_SSN) REFERENCES EMPLOYEE (SSN), FOREIGN KEY (DNO) REGERENCES DEPARTMENT (DNMUMBER));

DEPARTMENT: CREATE TABLE DEPARTMENT (DNAME VARCHAR (15) NOT NULL, DNUMBER INTEGER NOT NULL, MGR_SSN CHAR (9) NOT NULL, MGR_START_DATE DATE, PRIMARY KEY (DNUMBER), UNIQUE (DNAME), FOREIGN KEY (MGR_SSN) REFERENCE EMPLOYEE (SSN));

DEPT_LOCATIONS :CREATE TABLE DEPT_LOCATIONS (DNUMBER INTEGER NOT NULL, DLOCATION VARCHAR (15) NOT NULL, PRIMARY KEY (DNUMBER, DLOCATION), FOREIGN KEY (DNUMBER) REFERENCES DEPARTMENT (DNUMBER));

PROJECT :CREATE TABLE PROJECT (PNAME VARCHAR (15) NOT NULL, PNUMBER INTEGER NOT NULL, PLOCATION VARCHAR (15), DNUM INTEGER NOT NULL, PRIMARY KEY (PNUMBER), UNIQUE (PNAME), FOREIGN KEY (DNUM) REFERENCES DEPARTMENT (DNUMBER));

WORKS_ON :CREATE TABLE WORKS_ON (ESSN CHAR (9) NOT NULL, PNO INTEGER NOT NULL, HOURS DECIMAL (3, 1) NOT NULL, PRIMARY KEY (ESSN, PNO), FOREIGN KEY (ESSN) REFERENCES EMPLOYEE (SSN), FOREIGN KEY (PNO) REFERENCES PROJECT (PNUMBER));

DEPENDENT :CREATE TABLE DEPENDENT (ESSN CHAR (9) NOT NULL, DEPENDENT_NAME VARCHAR (15) NOT NULL, SEX CHAR, BDATE DATE, RELATIONSHIP VARCHAR (8), PRIMARY KEY (ESSN, DEPENDENT_NAME), FOREIGN KEY (ESSN) REFERENCES EMPLOYEE (SSN));

4) 1.Retrieve all attributes of any EMPLOYEE who works in DEPARTMENT number 5 :SELECT * FROM EMPLOYEE WHERE Dno=5;

2.Retrieve the name and address of all employees who work for the 'Research' department : SELECT Fname, Lname, Address FROM EMPLOYEE, DEPARTMENT WHERE Dname='Research' AND Dnumber= Dno;

3.For every project located in 'Stafford', list project num controlling dept number & dept manager's last name, address and birth date : SELECT Pnumber, Dnum, Lname, Address, Bdate FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE Dnum = Dnumber AND Mgr_ssn = ssn AND Plocation='Stafford';

4.For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor : SELECT E.Fname, E.Lname, S.Fname, S.Lname FROM EMPLOYEE AS E, EMPLOYEE AS S WHERE E.Super_ssn=S.Ssn;

5.Retrieve all the attributes of an EMPLOYEE and the attributes of the DEPARTMENT IN which her or she works for every employee of 'Research' department : SELECT * FROM EMPLOYEE, DEPARTMENT WHERE Dname='Research' AND Dno=Dnumber;

6.Retrieve the salary of every employee: SELECT ALL Salary FROM EMPLOYEE;

7.Retrieve all distinct salary values :SELECT DISTINCT Salary FROM EMPLOYEE;

8.Retrieve all employee whose address is in Houston, TeXas :SELECT Fname, Lname FROM EMPLOYEE WHERE Address LIKE '%Houston, TX%";

9. Find all employees who were born during the 1950; SELECT Fname, Lname FROM EMPLOYEE WHERE Bdate LIKE '1950%';

10.Retrieve all employees in department 5 whose salary is between 30,000 and 40,000 :SELECT * FROM EMPLOYEE WHERE (Salary BETWEEN 30000 AND 40000) AND Dno=5;

11.Show the resulting salaries if every employee working on the ProductX project is given a 10% raise:SELECT E.Fname, E.LNAME, 1.1 * E.Salary AS Increased_sal FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P WHERE E.SSN=W.ESSN AND W.Pno=P.Pnumber AND P.Pname='ProductX';

5)a.How the resulting salaries if every employee working on the 'Research' Departments is given a 10 percentage raise:SELECT E.ENO, E.ENAME, D.DNAME, 1.1 * E.SALARY AS "INC SALARY" FROM EMPLOYEE AS E, DEPARTMENT AS D WHERE E.DNO=D.DNUMBER AND D.DNAME="RESEARCH";

b. Find sum of the salaries of all employees of 'Accounts' dept , maximum salary, minimum salary & avg salary in this department :SELECT MAX (E.SALARY), MIN (E.SALARY), SUM (E.SALARY), AVG (E.SALARY) FROM EMPLOYEE E, DEPARTMENT D WHERE E.DNO=D.DNUMBER AND D.DNAME = "ACCOUNTS";

6)Retrieve the name of each employee Controlled by department number 5 (use EXISTS operator): SELECT E.ENAME FROM EMPLOYEE E WHERE EXISTS (SELECT D.DNUMBER FROM DEPARTMENT D WHERE E.DNO=D.DNUMBER AND E.DNO=5);

b. Retrieve name of each dept & num of employees working in each dept which has at least 2 employees: SELECT D.DNAME, COUNT (*) FROM EMPLOYEE E, DEPARTMENT D WHERE E.DNO=D.DNUMBER GROUP BY D.DNAME HAVING COUNT (*) > 2;

7)a. For each project, retrieve the project number, the project name, and the number of employee who work on that project. (Use GROUP BY) :SELECT P.PNUMBER, P.PNAME, COUNT (*) AS "NO_OF_EMP" FROM PROJECT P, WORKS_ON W WHERE P.PNUMBER=W.PNO GROUP BY P.PNUMBER, P.PNAME;

b. Retrieve name of employees who born in year 1990'. SELECT FNAME, BDATE FROM EMPLOYEE WHERE BDATE LIKE "1990%";

8) For each department that has more than five employees, retrieve the department number and number of employees who are making salary more than 40000' :SELECT D.DNAME, D.DNUMBER, COUNT (*) AS "NO_OF_EMP" FROM EMPLOYEE E, DEPARTMENT D WHERE E.DNO=D.DUMBER AND E.SALARY > 40000 AND D.DNUMBER IN (SELECT DNO FROM EMPLOYEE GROUP BY DNO HAVING COUNT (*) > = 5) GROUP BY D.DNO, D.DNAME;

9) For each project on which more than two employees work, retrieve the project number, project name and the number of employees who work on that project' :SELECT P.PNUMBER, P.PNAME, COUNT (*) AS "NO_OF_EMPLOYEES" FROM PROJECT P, WORKS_ON W WHERE P.PNUMBER = W.PNO GROUP BY P.PNUMBER, P.PNAME HAVING COUNT (*) > 2;

10)a. Create a view : CREATE VIEW EMP_DEPT AS (SELECT E.SSN, E.FNAME, E.SALARY, E.DEPTNO, D.DNAME FROM EMPLOYEE E, DEPARTMENT D WHERE E.DEPTNO= D.DNO);

b. Display all the rows of a view :SELECT * FROM EMP_DEPT;

c. Insert records into a view: INSERT INTO EMP_DEPT (SSN, FNAME, SALARY, DEPTNO) VALUES (10000009,'RAJESH', 90000, 5,'Research');

d. Display all the rows of a view :SELECT * FORM EMP_DEPT;

e. Drop view :DROP VIEW EMP_DEPT;

f. WITH CHECK OPTION 1. Let us create sample view on EMPLOYEE table with check option of salary less than 50000 in where condition

CREATE VIEW EMP_VIEW AS (SELECT SSN, FNAME, SALARY FROM EMPLOYEE WHERE SALARY <= 50000) WITH CHECK OPTION;

2. Display all the rows of a view:SELECT * FROM EMP_VIEW;

3. Insert a row where employee salary is less than 500000: INSERT INTO EMP EMP_VIEW (SSN, FNAME, SALARY) VALUES (101155555,'Ravi', 39000);

4. Display all the rows of views :SELECT * FROM EMP_VIEW;

5. Insert a row where employee salary is greater than 50000: INSERT INTO EMP_VIEW (SSN, FNAME, SALARY) VALUES (1012,"RAMU", 90999);

6. Drop view : DROP VIEW EMP_VIEW;

A2: **list of all the files in the current directory**

```
echo "Enter the directory name:";read dir;if [ -d "$dir" ];then cd "$dir" || exit;for file in *;do if [ -f "$file" ];then if [ -r "$file" ] && [ -w "$file" ] && [ -x "$file" ];then ls -ltr "$file";echo "File permissions are already granted.";else echo "No permissions for $file.";chmod 777 "$file";echo "Permissions granted.";ls -ltr "$file";fi;fi;done;else echo "Directory does not exist.";fi
```

A3: **accepts a list of file names as arguments, count and reports occurrence of each word**

```
if [ $# -lt 2 ];then echo "Enter more than one file name as argument";exit 1;fi;for i in $(cat "$@");do echo "word= $i";echo "-------";grep -c "$i" "$@";echo "-------";done
```

A4: **accepts one or more file name as arguments and converts all of them to uppercase**

```
if [ $# -lt 1 ];then echo "no arguments";exit 1;else for file in $@;do if [ ! -f $file ];then exit 1;else echo $file | tr '[ a-z ]' '[ A-Z ]';fi;done;fi
```

A5: **grep commands a. To select the lines from a file that has exactly 2 characters. b. that has more2 than one blank spaces.**

```
while true;do echo "Enter the file name:";read filename;echo "1. Select lines from the file that have exactly 2 characters.";echo "2. Select lines from the file that have more than 2 blank spaces.";echo "Enter your choice:";read ch;case $ch in 1) echo "Lines that have exactly 2 characters are:";grep -n '^...$' $filename;; 2) echo "Lines that have more than 2 blank spaces are:";grep '[[:space:]]\{2,\}' $filename > f1_result.txt;cat f1_result.txt;; *) exit;; esac;done
```

A6: **accepts two file names as arguments. Compare the contents. If they are same, then delete the second file.**

```
if [ $# -lt 1 ];then echo "no arguments";exit 1;else if [ ! -f $1 -o ! -f $2 ];then echo "file not existing";exit 1;else if cmp $1 $2;then echo $1 and $2 have identical contents;rm $2;echo "second file is removed";else echo $1 and $2 differ;fi;fi;fi
```

A7:**a. to count number of lines that do not contain vowels. b.count number of characters, words, lines**

```
while true;do echo "enter the filename";read file;echo "1. to count number of lines in a file that do not contain vowels";echo "2. to count number of characters words lines in a given file";echo "enter your choice";read ch;case $ch in 1)echo "no: of lines that do not contain vowels";grep -v '[aeiou]\+' $file;; 2)echo "to no: of characters";wc -c $file;echo "no: of lines";wc -l $file;echo "no: of words";wc -w $file;; *)exit;; esac;done
```

A8: **script to list all the files in a given directory**

```
echo "enter directory name";read dir;if [ -d $dir ];then echo "list of files in the directory";ls -l $dir | egrep '^-';else echo "enter proper directory name";fi
```

A9: **script to display list of users currently logged in.**

```
echo "Yourname:$(echo $USER)";echo "Current date & time:$(date)";echo "Currently logged on users: $(who)"
```

A10: **script to read three text files in the current directory and merge them into a single file**

```
echo "Enter first file name:";read f1;echo "Enter second file name:";read f2;echo "Enter third file name:";read f3;cat "$f1" "$f2" "$f3" > file4.txt;ls -l file4.txt;cat file4.txt
```

B1: **program to copy a file into another using system calls.**

```
#include<stdio.h>#include<unistd.h>#include<fcntl.h>int main(){char buf;int fd_one,fd_two;fd_one=open("p1.txt",O_RDONLY);if(fd_one==-1){printf("Error opening first_file\n");return 1;}fd_two=open("p4.txt",O_WRONLY|O_CREAT,S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);if(fd_two==-1){printf("Error opening or creating second_file\n");close(fd_one);return 1;}while(read(fd_one,&buf,1)){write(fd_two,&buf,1);}printf("Successful copy\n");close(fd_one);close(fd_two);return 0;}
```

B2: **program using system call: create, open, write, close, stat, fstat, lseek.**

```
#include<stdio.h>#include<stdlib.h>#include<sys/types.h>#include<sys/stat.h>#include<fcntl.h>#include<unistd.h>#define BUFSIZE 512 char buf1[]="Linux programming lab ";char buf2[]="program 1b ";char buf[BUFSIZE];struct stat buf3;int main(){int fd,status;if((fd=creat("t1.txt",0666))<0){perror("Creation error");exit(1);}if(write(fd,buf1,sizeof(buf1))<0){perror("Writing error");exit(2);}if(lseek(fd,4096,SEEK_SET)<0){perror("Positioning error");exit(3);}if(write(fd,buf2,sizeof(buf2))<0){perror("Writing error");exit(2);}if(stat("t1.txt",&buf3)<0){return 1;}printf("Information for file\n---------------------\nFile size:\t\t %ld bytes\nNumber of links:\t %ld\nFile inode:\t\t %ld\n",buf3.st_size,buf3.st_nlink,buf3.st_ino);fd=open("t1.txt",O_RDONLY);status=fstat(fd,&buf3);printf("\nStatus of the file is %d   ",status);if(fd==-1){printf("Error opening file\n");exit(1);}else{while(read(fd,&buf,1)){write(1,&buf,1);}}close(fd);}
```

B3: **create a child process and allow parent to display "parent" and child to display "child" on screen**

```
#include<stdio.h>#include<unistd.h>#include<sys/time.h>#include<sys/resource.h>void main(){int num;int procid[100]={0};int i;printf("Please enter number of processes to be created: \n");scanf("%d",&num);for(i=0;i<num;i++){procid[i]=fork();switch(procid[i]){case 0:while(1);break;default:break;} }printf("Parent process ID: %d\n",getpid());for(i=0;i<num;i++){printf("Child %d's ID is %d\n",i+1,procid[i]);}}
```

B4: **create a Zombie process**

```
#include<stdio.h>#include<unistd.h>int main(){pid_t ret_val;printf("The process ID is %d\n",getpid());ret_val=fork();if(ret_val==0){printf("The child process ID is %d\n",getpid());sleep(20);}else{printf("The parent process ID is %d\n",getpid());sleep(30);}return 0;}
```

B5: **implement inter process communication using pipes**

```
#include<stdio.h>#include<string.h>#include<stdlib.h>#include<unistd.h>#include<sys/time.h>#include<sys/resource.h>#include<sys/types.h>#include<sys/wait.h>int main(){char msg[80],buf[80];int p[2],pid,byread;pipe(p);printf("Reading fd is %d \n",p[0]);printf("Writing fd is %d \n",p[1]);pid=fork();if(pid>0){close(p[0]);printf("Enter the message: \n");byread=read(0,msg,80);write(p[1],msg,byread);wait(NULL);exit(0);}else{close(p[1]);read(p[0],buf,80);printf("The message read is:\n");printf("%s",buf);exit(0);}}
```

B6: **CPU scheduling algorithms a. SJF b. Round Robin**

A(sjf):

```
#include<stdio.h>int main(){int bt[20],p[20],wt[20],tat[20],i,j,n,tot_wt=0,tot_tat=0,pos,temp;float avg_wt,avg_tat;printf("Enter number of processes: ");scanf("%d",&n);printf("\nEnter burst time:\n");for(i=0;i<n;i++){printf("p %d: ",i+1);scanf("%d",&bt[i]);p[i]=i+1;}for(i=0;i<n;i++){for(j=i+1;j<n;j++){if(bt[i]>bt[j]){temp=bt[i];bt[i]=bt[j];bt[j]=temp;temp=p[i];p[i]=p[j];p[j]=temp;}}}wt[0]=0;for(i=1;i<n;i++){wt[i]=0;for(j=0;j<i;j++)wt[i]+=bt[j];tot_wt+=wt[i];}avg_wt=(float)tot_wt/n;printf("\nProcess\tBurst Time\tWaitingTime\tTurnaroundTime");for(i=0;i<n;i++){tat[i]=wt[i]+bt[i];tot_tat+=tat[i];printf("\np%d\t\t%d\t\t%d\t\t%d",p[i],bt[i],wt[i],tat[i]);}avg_tat=(float)tot_tat/n;printf("\n\nAverage waiting time = %f",avg_wt);printf("\nAverage turnaround time = %f\n",avg_tat);}
```

B(rr):

```
#include<stdio.h>int main(){int i,n,total=0,x,counter=0,qt;int tot_wt=0,tat=0,at[10],bt[10],burst_time[10];float avg_wt,avg_tat;printf("\nEnter total number of processes: ");scanf("%d",&n);x=n;for(i=0;i<n;i++){printf("\nEnter details of process[%d]\n",i+1);printf("Arrival time: ");scanf("%d",&at[i]);printf("Burst time: ");scanf("%d",&burst_time[i]);bt[i]=burst_time[i];}printf("\nEnter time quantum: ");scanf("%d",&qt);printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time\n");for(total=0,i=0;x!=0;){if(bt[i]<=qt&&bt[i]>0){total=total+bt[i];bt[i]=0;counter=1;}else if(bt[i]>0){bt[i]=bt[i]-
```

qt;total=total+qt;}if(bt[i]==0&&counter==1){x--;int turnaround_time=total-at[i];int waiting_time=turnaround_time-burst_time[i];printf("\nProcess[%d]\t\t%d\t\t%d\t\t%d",i+1,burst_time[i],turnaround_time,waiting_time);tot_wt=tot_wt+waiting_time;tat=tat+turnaround_time;counter=0;}if(i==n-1){i=0;}else if(at[i+1]<=total){i++;}else{i=0;}}avg_wt=(float)tot_wt/n;avg_tat=(float)tat/n;printf("\n\nAverage waiting time: %f",avg_wt);printf("\nAverage turnaround time: %f\n",avg_tat);return 0;}

B7: **file locking using semaphores**

#include<pthread.h>#include<stdio.h>#include<stdlib.h>#include<unistd.h>#include<semaphore.h>sem_t semaphore;FILE *fd;void* routine(void* args);int main(int argc,char* argv[]){pthread_t threadID;sem_init(&semaphore,0,1);system("rm -f f4");fd=fopen("f4","a");if(fd==NULL){perror("Failed to open file");return 1;}pthread_create(&threadID,NULL,&routine,NULL);sleep(1);sem_wait(&semaphore);printf("Hello from main function\n");fprintf(fd,"2222 writing from main function\n");sem_post(&semaphore);pthread_join(threadID,NULL);sem_destroy(&semaphore);fclose(fd);system("cat f4");return 0;}void* routine(void* args){sem_wait(&semaphore);printf("Hello from thread\n");fprintf(fd,"11111 writing from thread\n");sem_post(&semaphore);return NULL;}

B8: **producer-consumer system with two processes**

#include <stdlib.h> #include <unistd.h> #include <string.h> #include <semaphore.h> #include <stdio.h> #include <pthread.h> sem_t empty; sem_t full; int in = 0; int out = 0; int buffer[3]; pthread_mutex_t mutex; void *producer(void *pno) { int item; for (int i = 0; i < 3; i++) { item = rand() % 50; sem_wait(&empty); pthread_mutex_lock(&mutex); buffer[in] = item; printf("Producer: Insert item %d at %d\n", buffer[in], in); in = (in + 1) % 3; pthread_mutex_unlock(&mutex); sem_post(&full); } return NULL; } void *consumer(void *cno) { for (int i = 0; i < 3; i++) { sem_wait(&full); pthread_mutex_lock(&mutex); int item = buffer[out]; printf("Consumer: Remove item %d from %d\n", item, out); out = (out + 1) % 3; pthread_mutex_unlock(&mutex); sem_post(&empty); } return NULL; } int main() { pthread_t pro[3], con[3]; pthread_mutex_init(&mutex, NULL); sem_init(&empty, 0, 3); sem_init(&full, 0, 0); srand(time(0)); for (int i = 0; i < 3; i++) { pthread_create(&pro[i], NULL, &producer, NULL); pthread_create(&con[i], NULL, &consumer, NULL); } for (int i = 0; i < 3; i++) { pthread_join(pro[i], NULL); pthread_join(con[i], NULL); } pthread_mutex_destroy(&mutex); sem_destroy(&empty); sem_destroy(&full); return 0; }

B9: **inter process communication using shared memory system calls.**

A: #include<sys/ipc.h> #include<sys/shm.h> #include<stdio.h> int main() { key_t key = ftok("shmfile", 65); int shmid = shmget(key, 1024, 0666 | IPC_CREAT); char *str = (char*) shmat(shmid, (void*) 0, 0); printf("write data: \n"); fgets(str, 100, stdin); printf("data written in memory: %s\n", str); shmdt(str); return 0; }

B: #include<sys/ipc.h> #include<sys/shm.h> #include<stdio.h> int main() { key_t key = ftok("shmfile", 65); int shmid = shmget(key, 1024, 0666 | IPC_CREAT); char *str = (char*) shmat(shmid, (void*) 0, 0); printf("data read from memory = %s\n", str); shmdt(str); shmctl(shmid, IPC_RMID, NULL); return 0; }

B10: **a. Creating message queue b. Writing to a message queue c. Reading from a message queue**

A: #include<stdio.h> #include<string.h> #include<stdlib.h> #include<errno.h> #include<sys/types.h> #include<sys/ipc.h> #include<sys/msg.h> char mtext[2000]; int main(void) { int msgid; int len; key_t key = ftok("shmfile", 65); msgid = msgget(key, 0644 | IPC_CREAT); if (msgid == -1) { perror("msgget failed"); exit(1); } printf("msgqueue: ready to send msg.\n"); printf("write data: \n"); fgets(mtext, sizeof(mtext), stdin); len = strlen(mtext); if (msgsnd(msgid, &mtext, len, 0) == -1) { perror("msgsnd failed"); exit(1); } printf("message queue: done sending msg.\n"); return 0; }

B: #include<stdio.h> #include<stdlib.h> #include<errno.h> #include<sys/types.h> #include<sys/ipc.h> #include<sys/msg.h> #include<string.h> char mtext[2000]; int main(void) { int msgid; key_t key = ftok("shmfile", 65); msgid = msgget(key, 0644); printf("message queue: ready to receive message\n"); msgrcv(msgid, &mtext, sizeof(mtext), 0, 0); printf("received data: %s", mtext); printf("message queue: done receiving message\n"); return 0; }