Name : Anupam Kunwar
Reg : 19BCE1369

Lab-9

Q1. Deadlock Detection

Code :

```
#include <stdio.h>
static int mark[20];
int i, j, np, nr;

int main()
{
int alloc[10][10], request[10][10], avail[10], r[10], w[10];
printf("Enter the no of process: ");
scanf("%d", &np);
printf("Enter the no of resources: ");
scanf("%d", &nr);
printf("\n");
for (i = 0; i < nr; i++)
{
printf("Total Amount of the Resource R%d: ", i + 1);
scanf("%d", &r[i]);
}
printf("\nEnter the request matrix:\n");
for (i = 0; i < np; i++)
for (j = 0; j < nr; j++)
scanf("%d", &request[i][j]);
printf("\nEnter the allocation matrix:\n");
for (i = 0; i < np; i++)
for (j = 0; j < nr; j++)
scanf("%d", &alloc[i][j]);
for (j = 0; j < nr; j++)
{
avail[j] = r[j];
for (i = 0; i < np; i++)
{
avail[j] -= alloc[i][j];
}
}
for (i = 0; i < np; i++)
{
int count = 0;
for (j = 0; j < nr; j++)
{
if (alloc[i][j] == 0)
count++;
else
break;
}
if (count == nr)
```

```
mark[i] = 1;
}
for (j = 0; j < nr; j++)
w[j] = avail[j];
for (i = 0; i < np; i++)
{
int canbeprocessed = 0;
if (mark[i] != 1)
{
for (j = 0; j < nr; j++)
{
if (request[i][j] <= w[j])
canbeprocessed = 1;
else
{
canbeprocessed = 0;
break;
}
}
if (canbeprocessed)
{
mark[i] = 1;

for (j = 0; j < nr; j++)
w[j] += alloc[i][j];
}
}
}
int deadlock = 0;
for (i = 0; i < np; i++)
if (mark[i] != 1)
deadlock = 1;

if (deadlock)
printf("\nDeadlock detected\n");
else
printf("\nNo Deadlock possible\n");
}
```

Output :

```
piratepanda@SastaPC:~/Documents/oslab/week9$ ./deadDetect.out
Enter the no of process: 3
Enter the no of resources: 4

Total Amount of the Resource R1: 2
Total Amount of the Resource R2: 5
Total Amount of the Resource R3: 4
Total Amount of the Resource R4: 3

Enter the request matrix:
0 2 0 1
0 1 0 2
0 1 0 1

Enter the allocation matrix:
1 1 0 0
1 0 1 0
1 2 0 0

Deadlock detected
piratepanda@SastaPC:~/Documents/oslab/week9$
```

.

Q2. Multi-Threading

Code :

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define N 5

void *worker_thread(void *arg)
{
printf("Thread #%ld\n", (long)arg);
pthread_exit(NULL);
}

int main()
{
pthread_t my_thread[N];
long id;
for (id = 1; id <= N; id++)
{
int ret = pthread_create(&my_thread[id], NULL, &worker_thread, (void *)id);
if (ret != 0)
```

```
{
printf("Error: pthread_create() failed\n");
exit(EXIT_FAILURE);
}
}
pthread_exit(NULL);
}
```

Output :

```
piratepanda@SastaPC:~/Documents/oslab/week9$ ./a.out
Thread #1
Thread #2
Thread #3
Thread #4
Thread #5
piratepanda@SastaPC:~/Documents/oslab/week9$
```

.

Q3. Binary Semaphores.

Code :

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t mutex;

void display(void *arg){
printf("\nThread #%ld is inside critical section\n",(long)arg);
}

void *thread(void *arg)
{
//wait
sem_wait(&mutex);
printf("\nEntered..\n");

//critical section
display(arg);
sleep(4);

//signal
printf("\nJust Exiting...\n");
sem_post(&mutex);
}

int main()
{
```

```
sem_init(&mutex, 0, 1);
pthread_t t1, t2;
pthread_create(&t1, NULL, thread,(void*)1);
sleep(2);
pthread_create(&t2, NULL, thread,(void*)2);
pthread_join(t1, NULL);
pthread_join(t2, NULL);
sem_destroy(&mutex);
return 0;
}
```

Output :

```
piratepanda@SastaPC:~/Documents/oslab/week9$ gcc Semaphore.c -lpthread -lrt
piratepanda@SastaPC:~/Documents/oslab/week9$ ./a.out

Entered..

Thread #1 is inside critical section

Just Exiting...

Entered..

Thread #2 is inside critical section

Just Exiting...
piratepanda@SastaPC:~/Documents/oslab/week9$
```