Week 6: OS lab. Fasilu

1 (a)

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

void sort(int p[], int d[] int b[], int pt[], int n){
    int temp;
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            if(d[j] < d[i]){
                temp = d[j];
                d[j] = d[i];
                d[i] = temp;

                temp = pt[i];
                pt[i] = pt[j];
                pt[j] = temp;

                temp = b[j];
                b[j] = b[i];
                b[i] = temp;

                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }
}
```

```c
int gcd (int a, int b) {
    int r;
    while (b>0) {
          r = a % b;
          a = b;
          b = r;
    }
    return a;
}
int lcmul(int p[], int n) {
     int lcm = p[0];
     for(int i = 1; i<n ;i++) {
            lcm = lcm* p[i] /gcd (lcm, p[i]);
     }
    return lcm;
}
void main() {
   int n;
   printf("Enter the no. of processors");
   scanf("%d", &n);
   int p[n], b[n], pt[n], d[n], rem[n];
   printf("Enter the CPU burst time");
   for(int i=0; i<n ;i++) scanf("%d", &b[i]);
   printf("Enter the deadlines");
   for(int i=0 ; i<n ; i++)  scanf("%d", &d[i]);
   printf("Enter the time period\n");
                              d("%d", &pt[i]);
```

```
for(int i=0;i<n;i++) p[i]=i+1;

sort(p,d,b,pt,n);
int l = lcmul(pt,n);
printf("\n Earlier Deadline Scheduling \n");
printf("PID\t Burst \t Deadline \t Period\n");

for(int i=0;i<n;i++)
        printf("%d\t%d\t \t %d \t \t %d \n"
                    , p[i], b[i], d[i], pt[i]);

print("Scheduling occurs for %d ms \n",l);

int time =0, nextD[n];
for(int i=0;i<n;i++){
        nextD[i] = d[i];
        rem[i] = b[i];
}

while (time<l){
        for(int i=0; i<n;i++){
                if(time %d pt[i]==0 &&
                        time !=0){
                nextD[i]=time+d[i];
                }  rem[i]=b[i];
        }

int minD = l+1, taskToExecute = -1;
for(int i=0 ; i<n ;i++){
        if(rem[i]>0 && nextD[i] < minD){
                minD = nextD[i];
                taskToExecute~~ß~~ = i;
        }
}

if(taskToExecute != -1){
        printf("%dms : Task %d is running\n",
                        t, p[taskToExecute]);
        rem[taskToExecute]--;

}
else {
        printf("%dms: CPU is idle \n", time);

}
time ++;

}

}
```

Enter no. of processor :3

Enter CPU burst time :

3

2

2

Enter deadline:

7

4

8.

Enter time period

20

5

Earliest deadline scheduling:

PID	Burst	Deadline	Period
2	2	4	5
1	3	7	10
3	2	8	10

Scheduling Occurs 20 ms.

0ms : Task 2 running

2ms : Task 1 running

5ms : Task 3 running

7ms : Task 2 running

9 : CPU is idle

10ms : Task 2 running.

12 ms : Task 3 running

14ms : idle

15 ms : Task 2 running.

17 : CPU idle

~~Rate monotonic schedule~~

Rate monotonic scheduling:

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

void swap(int *a, int* b){
        int temp = *a;
        *a = *b;
        *b = temp; }
```

```
void sort(int p[], int b[], int pt[], int n){
    int temp=0;
    for(int i=0;i<n;i++){
            for(int j=i;j<n;j++){
                  if(pt[j]<pt[i])
            {
                  swap(&pt[i],&pt[j]);
                  swap(&b[i],&b[j]);
                  swap(&p[i],&p[j]);
            }
      }
} }
int gcd(int a,int b){
    int r;
    while(b>0){
         r=a%b;
         a=b;
         b=r;
    }
   return a;
}
```

~~lcmul~~ (int p[], int n){

```
int lcm=p[0];
for(int i=1;i<n;i++)
{
    lcm = (lcm* p[i])/gcd(lcm,p[i]);
}
```

```
void main (){
    int n;
    printf("Enter no. of processor");
    scanf("%d",&n);
    int p[n], b[n], pt[n], rem[n];
     printf("Enter the CPU burst time\n");
    for (int i=0; i<n; i++)
    {
          scanf ("%d", &b[i]);
          rem[i] = b[i];
    }
   print("Enter time period of n");
  for(int i=0; i<n; i++)
          scanf("%d", &pt[i]);
  for(int i=0; i<n; i++)
         p[i]=i+1;
 sort (p, b, pt, n);
 int l = lcmul(pt, n);
  printf("LCM=%d\n", l);
 printf("Rate monotonic scheduling\n");
 printf(" PID\tBurst\tPeriod\n");
for(int i=0; i<n; i++)
         printf("%d\t%d\t%d\n",
                    p[i], b[i], pt[i]);
```

```
doub
for(int i=0;i
    sum+= (doub
double rhs = n*(pow(2.0,(1.0/
printf("\n%lf <=%lf\n", sum, rhs,
      (sum<=rhs) ?"true" : "false");

if (sum > rhs)
      exit(0);

printf(" scheduling occurs for %dms\n",l);
int time=0, prev=0; x=0;
while (time <l){
    int f=0;
    for(int i=0; i<n; i++){
          if (time %d pt[i]==0)
                rem[i]= b[i];
          ~~if~~ (rem[i] >0){
              if (prev != p[i])
              {
                  printf(".%dms onwards
                        proc %d running
                        \n",time,
                        ~~proc~~[i]);
                  prev = p[i];
              }
              rem[i]--;
              f=1;
              break;
    if(!f){
        if(x!=1)
        {
            printf("%dms onwards:
                      CPU-is idle\n",tim);
            x=1;
        }
    }
    time++;
  }
}.
```

Output:

```
Enter no. of processor: 2
Enter the CPU burst time:
20
25
Enter time periods:
50
100
LCM=100
Rate monotonic scheduling:
PID       Burst      period
 1         20          50
 2         25          100
0.65000 <=0.828427 => true
Scheduling Occurs for 100ms
0ms onwards: processor 1 running
20ms onwards: processor 2 running
50ms onwards: processor 1 running
```

some a

1© Propo

#inclu

incl

int r

some

3.① Proportional Scheduling

```
#include <stdlib.h>
#include <time.h>
int main(){
     srand(time(NULL));
     int n;
     printf("Entu no. g procuorn");
     scanf("%d", &n);
     int p[n], t[n], cum[n], m(n);
     int c=0; int total=0, count=0;

     printf("Entu tickets g proccessr\n");

    for(int i=0; i<n; i++){
           scanf("%d", &t(i));
           c+=t[i];
           cum [i]=c;
           p(i) =i+1;
           m(i) =0;
           total += t(i);

    }
  while (count <n)
        int wt = rand()%total;
        for(int i=0; i<n; i++)
        {
                printf("The winning number is
                        %d and winning participant
                    is %d \n", wt, p[i]);
                m[i]=1;
                count=1;
          }   }
    }
    printf("\nProbabilities:\n");
    for (int i=0; i<n; i++)
    {
        printf("The probability of p%d
                winning: %.2f %%\n", p[i],
                ((double)t[i]/total * 100));
        }
    }
```

Output:

```
Enter no. of processors :3
Enter tickets of the processes;
10
20
30
~~The winning number probability q +~~
Probabilities
The probability of p1 winning : 16.67
The probability of P2 winning : 33.33
The probability of P3 winning : 50.00.
```

2) Write a C program to simulate producer consumer problem using semaphore

```
#include <stdio.h>
int mutex=1,h=0,empty=3,x=0;
int main()
{
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);
    printf("\n producer \n 2 consumer \3 exit ");
    while(1)
    {
        printf("Enter your choice:\n");
        scanf("%d",&n);
        switch(n)
        {
          case 1: if((mutex==1)&&(empty!=0))
                  producer();
                  else
                  printf("Buffer is full");
                    break
          case 2: if (mutex==1)&&(h!=0))
                      consumer
                    else
                        printf("Buffer is empty");
                    break;
          case 3: exit(0);
                    break;
        }
    }
    return 0;
}
```

```
int wait(int s){
        return(--s);
}
int signal(int s){
        return(++s);
}
void producer(){
    mutex = wait(mutex);
      n = signal(n);
    empty = wait(empty);
      x++;
    printf("producer produce the item %d",x);
    mutex = signal(mutex);
}
void consumer(){
    mutex = wait(mutex);
     n = wait(n);
    empty = signal(empty);
    printf("Consumer consume item %d",x);
    x--;
    mutex= signal(mutex);
}
```

output:

```
1) producer      2. consumer      3. exit.
Enter your choice: 1
producer produce item 1
Enter your choice 1
producer produce item 2
Enter your choice: 1
producer produce item 3
Buffer is full
Enter your choice: 2
consumer consume item 3
```

```
Futu
consum consume item-2
Ente your choice : 2
consume choice item 2
Buffer is empty
```

6. Write a C program to simulate the concept of dining-philosophers problem

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

# define NUM-PHIL 3
sem-t forks[NUM-PHIL)
pthread-t p[NUM-PHIL);
void * philosopher (void *arg) {
        int id = *((int*) arg);
        int left-fork = id ;
        int right-fork = (id + 1) % NUM_PHIL ;
      while (1) {
         printf("philosopher %d is thinking\n",
                 id);
         sleep (rand() % 3 + 1 );
         printf("philosopher %d is hungor and
                 trying to pick forks\n",
```