Operating System
ID: 519021911248
Name: ZhuoHao Li

Homework 1

2022-4-1

1 Part1

Solution.1

The last digit of pi is 5
The last digit of pi is 5

Solution.2

Hello World
:0
Hello World
:90210

Solution.3

0
1
2
3
(return all files'names in `pwd`)

Solution.4

If execution goes well, 5 processes will be created, 2 threads will be created.

Solution.5

B.

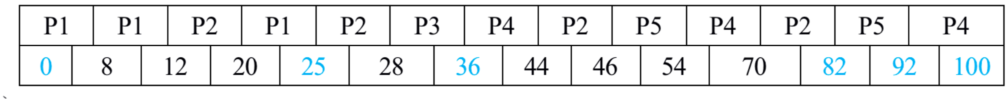
Solution.6

I am the child

2 Part2

Solution.1

(a)



P1 | P1 | P2 | P1 | P2 | P3 | P4 | P2 | P5 | P4 | P2 | P5 | P4

0 | 8 | 12 | 20 | 25 | 28 | 36 | 44 | 46 | 54 | 70 | 82 | 92 | 100

1: MFQS Gantt

(b)

Text switch: 14-6=8 times.

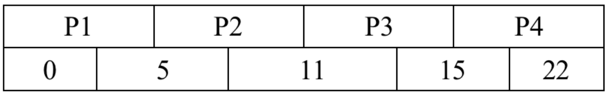
(c)

AWT: $[(25-0-17)+(82-12-25)+(36-8-28)+(100-36-32)+(92-46-18)]/5=22.6$

ATT: $[(25-0)+(82-12)+(36-28)+(100-36)+(92-46)]/5=42.6$

Solution.2

(a)



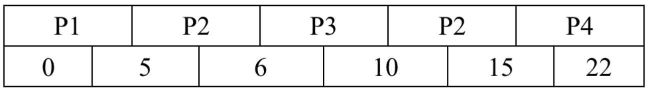
P1	P2	P3	P4	
0	5	11	15	22

2: FCFS Gantt

AWT: [(5 – 0 – 5) + (11 – 3 – 6) + (15 – 6 – 4) + (22 – 9 – 7)]/4 = 3.25

AAT: [(0 – 0) + (5 – 3) + (11 – 6) + (15 – 9)]/4 = 3.25

(b)



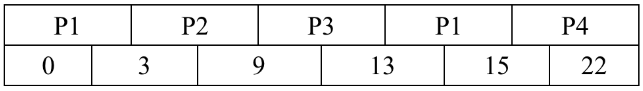
P1	P2	P3	P2	P4	
0	5	6	10	15	22

3: SRTF Gantt

AWT: [(5 – 0 – 5) + (15 – 3 – 6) + (10 – 6 – 4) + (22 – 9 – 7)]/4 = 3

AAT: [(0 – 0) + (5 – 3) + (6 – 6) + (15 – 9)]/4 = 2

(c)



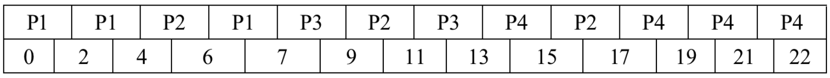
P1	P2	P3	P1	P4	
0	3	9	13	15	22

4: Priority Gantt

AWT: [(15 – 0 – 5) + (9 – 3 – 6) + (13 – 6 – 4) + (22 – 9 – 7)]/4 = 4.75

AAT: [(0 – 0) + (3 – 3) + (9 – 6) + (15 – 9)]/4 = 2.25

(d)



P1	P1	P2	P1	P3	P2	P3	P4	P2	P4	P4	P4	
0	2	4	6	7	9	11	13	15	17	19	21	22

5: RR Gantt

AWT: [(7 – 0 – 5) + (17 – 3 – 6) + (13 – 6 – 4) + (22 – 9 – 7)]/4 = 4.75

AAT: [(0 – 0) + (4 – 3) + (7 – 6) + (13 – 9)]/4 = 1.5

Solution.3

Pseudo code:

```
// Semaphore Difinition
semaphore mutex = 1;    // R/W buffer mutual exclusion
semaphore empty = N;    // Empty elements counter
semaphore odd = 0;      // Odd elements availble
semaphore even = 0;     // Even elements availble
int number;            // Store elements

Process1:
while (true) {
  number = produce();
  P(empty);  // take an EMPTY
  P(mutex);  // P(mutex) and V(mutex) protect a process execution
  put();
  V(mutex);
   if (number % 2) begin
    V(odd); // release an ODD
   else
    V(even); // release an EVEN
  end
}

Process2:
while (true) {
  P(odd); // take an ODD
  P(mutex);
  getodd();
  V(mutex);
  V(empty); // release an EMPTY
  countodd();
}

// P3 is the same as P2

P3:
while (true) {
  P(even);
  P(mutex);
  geteven();
  V(mutex);
  V(empty);
  counteven();
}
```

process.cc

Solution. 4

2 threads parallism

Just divide the original process to 1) Odd rows multilpy 2) Even rows multiply. They are named as `Thread1()` and `Thread2()`. Main thread needs to wait untill both are finished.

```
# include <bits/stdc++.h>
// row, col
void Thread1()
{
  int i,j,m;
  for(i=0;(i<row);i++){
    for(j=0;j<c;j++){
      for(m=0;m<n;m++){
        if(i%2==0) // divide threads by MOD2
        result[i][j]+=a[i][m]*b[m][j];
    }
  }
}

void Thread2()
{
  int i,j,m;
  for(i=0;(i<row);i++){
    for(j=0;j<c;j++){
      for(m=0;m<n;m++){
        if(i%2==0)
        result[i][j]+=a[i][m]*b[m][j];
      }
    }
  }

pthread_create(&tid[0],NULL,(void*)Thread1,NULL);
pthread_join(tid[0],NULL);
pthread_create(&tid[1],NULL,(void*)Thread2,NULL);
pthread_join(tid[1],NULL);
```

2 thread.cc

multiple threads parallism

A($M \times K$) and B($K \times N$) are 2 matrixes. A×B=C ($M \times N$. For each element of C, it can be an independent thread, which will generate M × N threads and pass the values of row i and column j to each thread. The thread uses the values of rows and columns to calculate the corresponding elements by $C_{i,j} = \sum_{n=1}^{K} A_{i,n} \times B_{n,j}$. When all threads end, the main thread can output matrix C, so in the code, the main thread needs to wait for all worker threads to complete the work. These matrices are declared as global data by default so that each worker thread can access matrices A, B, and C.

```
# include <bits/stdc++.h>
void Thread(int *p){
  int row=p[0];
  int col=p[1];
  int res=0;
  int l;
  for(l=0;l<n;l++){
    res+=a[row][l]*b[l][col];
    result[row][col]=res;
  }
```

```
    }

    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            pass[i][j][0]=i;
            pass[i][j][1]=j;
            r=pthread_create(&thid[i][j],NULL,(void*)Thread,pass[i][j]);
            pthread_join(thid[i][j],NULL);
        }
    }

```

multiple thread.cc