

Administrative matter

- Homework #5
 - Due Now!!!
- Quiz 2
 - Tuesday 11/26
- No lecture on 11/28
 - Happy Thanksgiving
- Final examination review...etc
 - Thursday 12/5
- Final examination
 - Tuesday 12/10, 8-11AM



1

CS122A: Embedded System Design, Fall 01

N-Version Programming vs Recovery Blocks

- Static (NV) versus dynamic redundancy (RB)
- Design overheads
 - both require alternative algorithms
 - NV requires driver, RB requires acceptance test
- Runtime overheads
 - NV requires N * resources, RB requires establishing recovery points
- Diversity of design
 - both susceptible to errors in initial requirements
- Error detection
 - vote comparison (NV) versus acceptance test(RB)
- Atomicity
 - NV vote before it outputs to the environment
 - RB must be structure to only output after passing of an acceptance test

2

CS122A: Embedded System Design, Fall 01

Dynamic Redundancy and Exceptions

- An exception can be defined as the occurrence of an error
- raising (or signally or throwing)
 - Bringing an exception to the attention of the invoker
- handling (or catching)
 - The invoker's response
- Exception handling is a forward error recovery
 - there is no roll back to a previous state
 - control is passed to the handler
 - However, the handler can be used to provide backward ER



CS122A: Embedded System Design, Fall 01

3

Exceptions

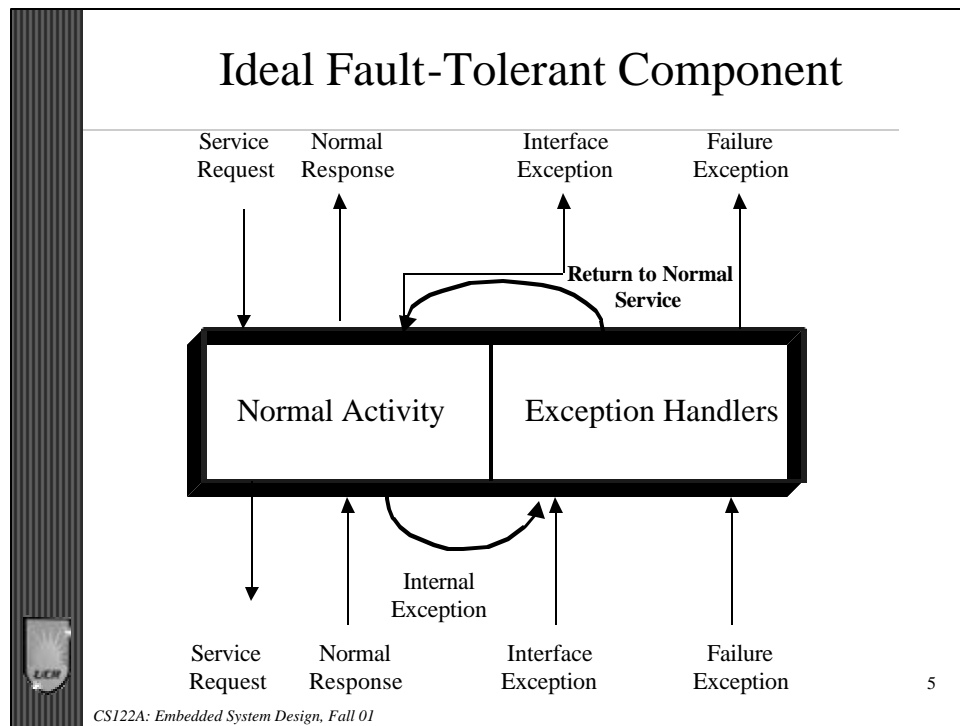
Exception handling can be used to:

- cope with abnormal conditions in the environment
- enable program design faults to be tolerated
- provide general-purpose error-detection and recovery



CS122A: Embedded System Design, Fall 01

4



5

- ### Safety and Reliability
- **Safety**
 - freedom from those conditions that can
 - cause death, injury, occupational illness, damage to (or loss of) equipment (or property), or environmental harm
 - **By this definition, most systems which have an element of risk associated with their use as unsafe**
 - **Reliability:**
 - a measure of the success with which a system conforms to specification
- CS122A: Embedded System Design, Fall 01

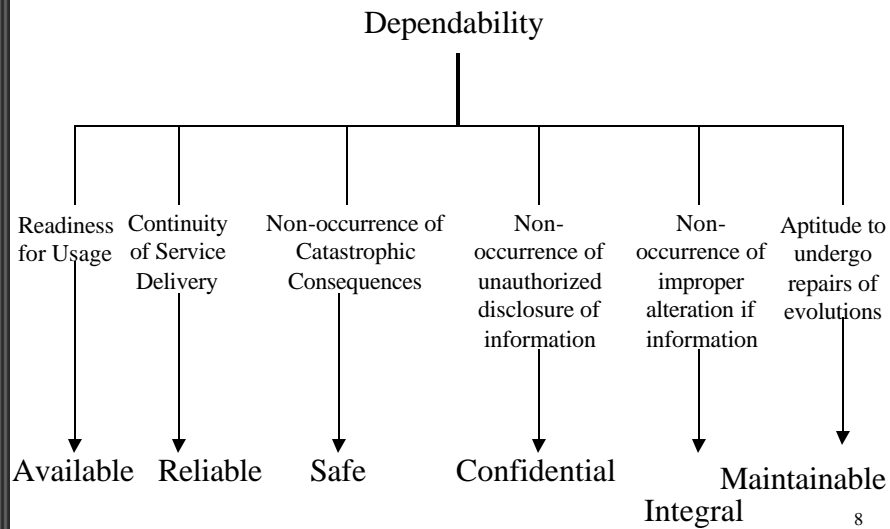
6

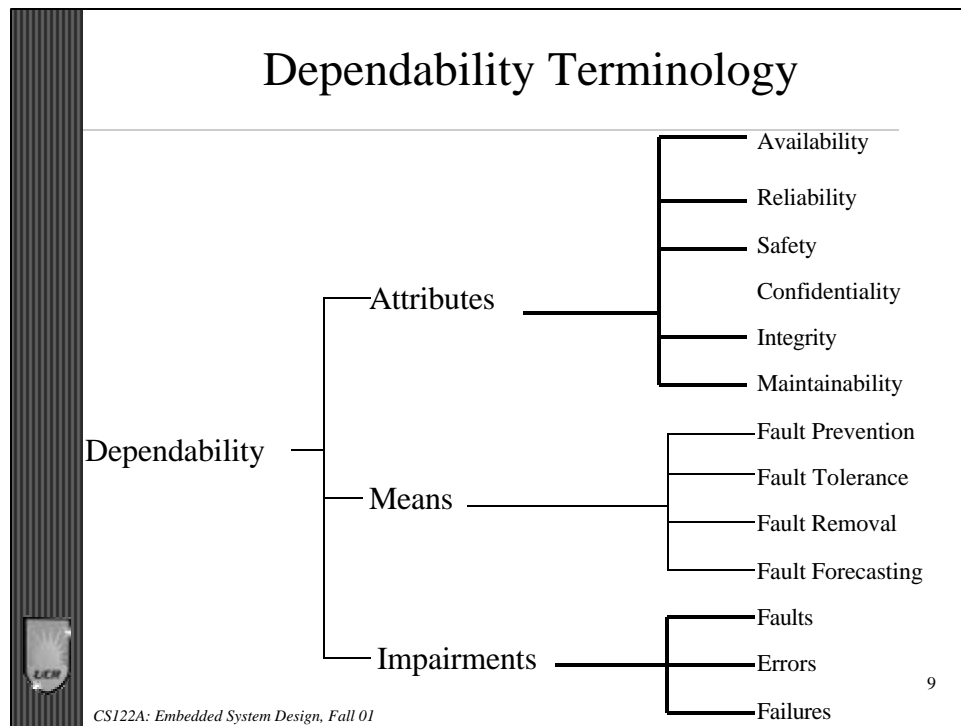
Safety Vs. Reliability

- Example
 - Measures which increase the likelihood of a weapon firing when required may well increase the possibility of its accidental detonation
- In many ways, the only safe airplane is one that never takes off however, it is not very reliable.



Aspects of Dependability





Scheduling

- Goal
 - To understand the role that scheduling and schedulability analysis plays in predicting that real-time applications meet their deadlines
- Topics
 - Simple process model
 - The cyclic executive approach
 - Process-based scheduling
 - Utilization-based schedulability tests
 - Response time analysis for FPS and EDF
 - Worst-case execution time
 - Sporadic and aperiodic processes
 - Process systems with $D < T$
 - Process interactions, blocking and priority ceiling protocols
 - An extendible process model
 - Dynamic systems and on-line analysis

10

CS122A: Embedded System Design, Fall 01

Scheduling

- In general, a scheduling scheme provides two features:
 - An algorithm for ordering the use of system resources (
 - in particular the CPUs
 - Predicting the worst-case behaviour of the system
 - when the scheduling algorithm is applied
- The prediction
 - used to confirm the temporal requirements of the application



Simple Process Model

- The application is assumed to consist of a fixed set of processes
- All processes are periodic, with known periods
- The processes are completely independent of each other
- System's overheads, context-switching times,...,etc are ignored
 - i.e, assumed to have zero cost
- All processes have a deadline equal to their period
 - each process must complete before it is next released
- All processes have a fixed worst-case execution time



Standard Notation

B	Worst-case blocking time for the process (if applicable)
C	Worst-case computation time (WCET) of the process
D	Deadline of the process
I	The interference time of the process
J	Release jitter of the process
N	Number of processes in the system
P	Priority assigned to the process (if applicable)
R	Worst-case response time of the process
T	Minimum time between process releases (process period)
U	The utilization of each process (equal to C/T)
a-z	The name of a process



CS122A: Embedded System Design, Fall 01

13

Cyclic Executives

- Common way of implementing hard real-time systems
- The design is concurrent
 - but the code is produced as a collection of procedures
- Procedures are mapped onto a set of minor cycles
- Complete schedule (or major cycle) consists of minor cycles
- Minor cycle dictates the minimum cycle time
- Major cycle dictates the maximum cycle time

Has the advantage of being fully deterministic



CS122A: Embedded System Design, Fall 01

14

Consider Process Set

Process	Period , T	Computation Time , C
a	25	10
b	25	8
c	50	5
d	50	4
e	100	2



CS122A: Embedded System Design, Fall 01

15

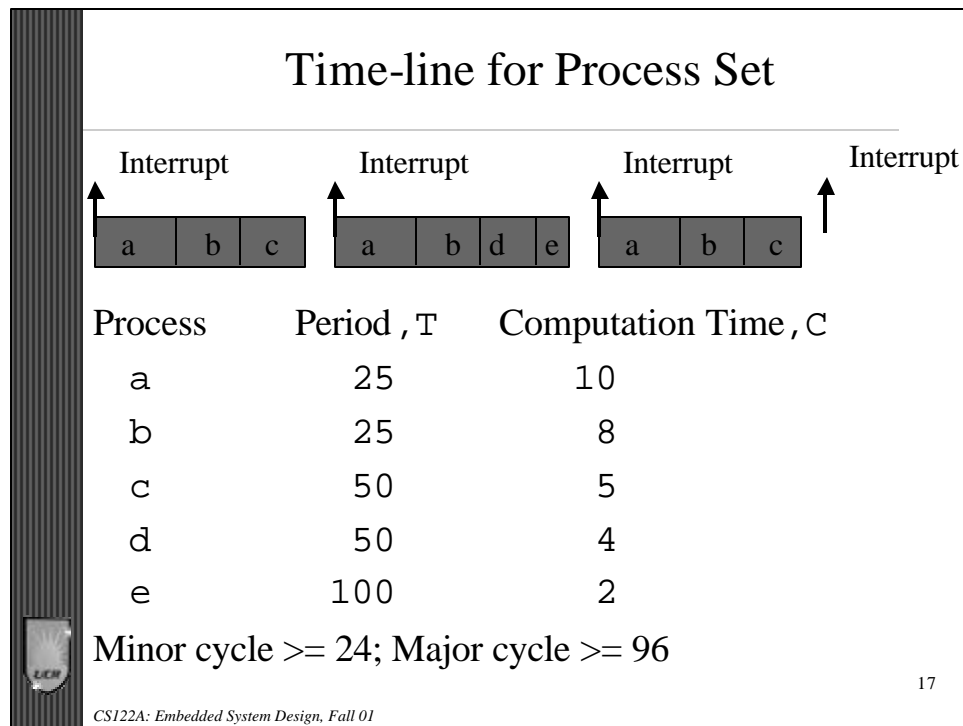
Cyclic Executive

```
loop  
  wait_for_interrupt;  
  procedure_for_a; procedure_for_b; procedure_for_c;  
  wait_for_interrupt;  
  procedure_for_a; procedure_for_b; procedure_for_d;  
  procedure_for_e;  
  wait_for_interrupt;  
  procedure_for_a; procedure_for_b; procedure_for_c;  
  wait_for_interrupt;  
  procedure_for_a; procedure_for_b; procedure_for_d;  
end loop;
```



CS122A: Embedded System Design, Fall 01

16



Properties

- No actual processes exist at run-time
 - Each minor cycle is just a sequence of procedure calls
- The procedures share a common address space
 - Can pass data between themselves
 - This data does not need to be protected (e.g. semaphore)
 - Concurrent access is not possible

18

CS122A: Embedded System Design, Fall 01

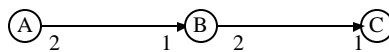
Problems with Cycle Executives

- The difficulty of incorporating processes with long periods
 - the major cycle time is the maximum period that can be accommodated
- Sporadic activities are difficult (impossible!) to incorporate
- The cyclic executive is difficult to construct and maintain
 - it is a NP-hard problem
- Process with a sizable computation time will need to be split into a fixed number of fixed sized procedures
- More flexible scheduling methods are difficult to support
- Determinism is not required, but predictability is

19

CS122A: Embedded System Design, Fall 01

Single Appearance Schedules



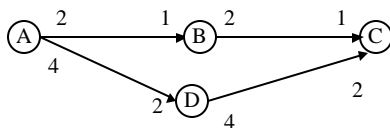
- ABCBCCC
 - Not SAS, require buffer $2+3=5$
- A(2B)(4C)
 - SAS, but require buffer $2+4=6$ (same as ABBCCCC)
- A(2 B(2 C))
 - SAS, and require buffer $2+2=4$ (same as ABCCBCC)

```
run A;
For (I=0; I<2; I++){
    run B;
    for(j=0;j<2;j++){
        run C;
    }
}
```

20

CS122A: Embedded System Design, Fall 01

Single Appearance Schedules (2)



- ABDCBDCCC
 - Not SAS, require buffer $6+9=15$
- A(2BD)(4C)
 - SAS, but require buffer $6+12=18$ (same as ABDBDCCCC)
- A(2 BD(2 C))
 - SAS, and require buffer $6+6=12$ (same as ABDCCBDCC)



CS122A: Embedded System Design, Fall 01

21

Process-Based Scheduling

- Scheduling approaches
 - Fixed-Priority Scheduling (FPS)
 - Earliest Deadline First (EDF)
 - Value-Based Scheduling (VBS)



CS122A: Embedded System Design, Fall 01

22

Fixed-Priority Scheduling (FPS)

- This is the most widely used approach
- Each process has a fixed, static, priority which is compute pre-run-time
- The runnable processes are executed in the order determined by their priority
- In real-time systems, the “priority” of a process is derived from its temporal requirements
 - not its importance to the correct functioning of the system or its integrity



Earliest Deadline First (EDF) Scheduling

- The runnable processes are executed in the order determined by the absolute deadlines of the processes
- The next process to run being the one with the shortest (nearest) deadline
- It is usual to know the relative deadlines of each process (e.g. 25ms after release)
- Absolute deadlines are computed at run time and hence the scheme is described as dynamic



Value-Based Scheduling (VBS)

- If a system can become overloaded then the use of simple static priorities or deadlines is not sufficient; a more adaptive scheme is needed
- This often takes the form of assigning a value to each process and employing an on-line value-based scheduling algorithm to decide which process to run next



Preemption and Non-preemption

- A high-priority process may be released during the execution of a lower priority one
- preemptive schemes
 - an immediate switch to the higher-priority process
- non-preemption schemes, the lower-priority process will be allowed to complete before the other executes
- Preemption enable higher-priority processes to be more reactive
- Deferred preemption or cooperative dispatching
 - strategies allow a lower priority process to continue for bounded time
- Schemes such as EDF and VBS can be either



FPS and Rate Monotonic Priority Assignment

- Each process is assigned a unique priority based on its period
- The shorter the period, the higher the priority
- I.e, for two processes i and j ,

$$T_i < T_j \Rightarrow P_i > P_j$$
- This assignment is optimal
- if any process set can be scheduled (using pre-emptive priority-based scheduling) with a fixed-priority assignment scheme
 - then the given process set can also be scheduled with a rate monotonic assignment scheme
- Note, priority 1 is the lowest (least) priority



27

CS122A: Embedded System Design, Fall 01

Example Priority Assignment

Process	Period, T	Priority, P
a	25	5
b	60	3
c	42	4
d	105	1
e	75	2



28

CS122A: Embedded System Design, Fall 01

Utilization-Based Analysis

- For D=T task sets only
- A simple sufficient but not necessary schedulability test exists

$$U \equiv \sum_{i=1}^N \frac{C_i}{T_i} \leq N (2^{1/N} - 1)$$

$$U \leq 0.69 \text{ as } N \rightarrow \infty$$



CS122A: Embedded System Design, Fall 01

29

Utilization Bounds

N	Utilization bound
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%

Approaches 69.3% asymptotically




CS122A: Embedded System Design, Fall 01

30


Process Set A

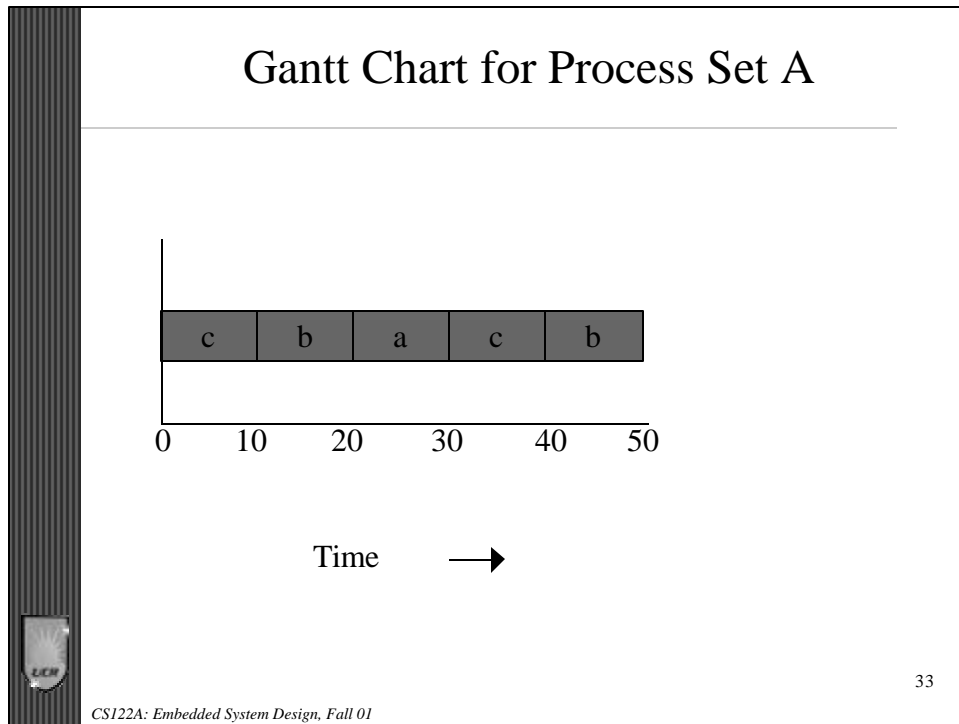
Process	Period T	ComputationTime C	Priority P	Utilization U
a	50	12	1	0.24
b	40	10	2	0.25
c	30	10	3	0.33

- The combined utilization is 0.82 (or 82%)
- This is above the threshold for three processes (0.78)
 - hence, this process set fails the utilization test


CS122A: Embedded System Design, Fall 01
31

Time-line for Process Set A


CS122A: Embedded System Design, Fall 01
32



Process Set B

Process	Period T	ComputationTime C	Priority P	Utilization U
a	80	32	1	0.400
b	40	5	2	0.125
c	16	4	3	0.250

- The combined utilization is 0.775
- This is below the threshold for three processes (0.78)
 - hence, this process set will meet all its deadlines

34

CS122A: Embedded System Design, Fall 01

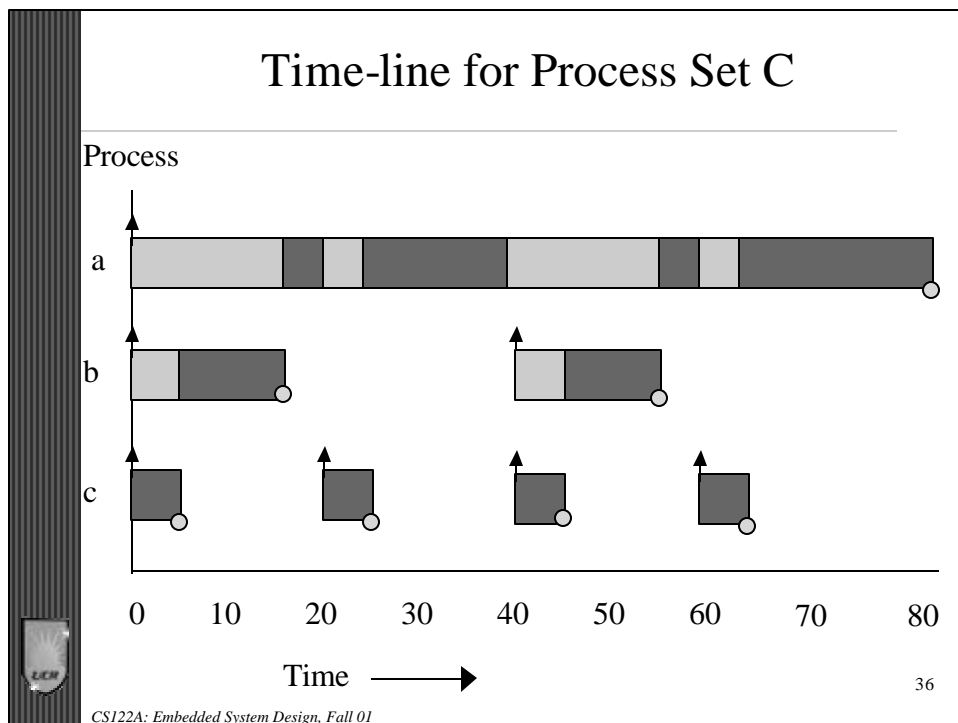
Process Set C

Process	Period T	ComputationTime C	Priority P	Utilization U
a	80	40	1	0.50
b	40	10	2	0.25
c	20	5	3	0.25

- The combined utilization is 1.0
- This is above the threshold for three processes (0.78)
 - but the process set will meet all its deadlines

35

CS122A: Embedded System Design, Fall 01



Criticism of Utilization-based Tests

- Not exact
- Not general
- BUT it is O(N)

The test is said to be sufficient but not necessary



Utilization-based Test for EDF

$$\sum_{i=1}^N \frac{C_i}{T_i} \leq 1 \quad \text{A much simpler test}$$

- High utilizations than FPS
- Harder to implement than FPS
- Requires a more complex run-time system, I.e. higher overhead
- Processes without deadline
 - FPS: Just set deadline to infinite, and give it arbitrary (low) priority
 - EDF: some arbitrary large deadline, a little more awkward
- During overload situations
 - FPS is more predictable;
 - Low priority process miss their deadlines first
 - EDF is unpredictable
 - domino effect can occur in which many processes miss deadlines

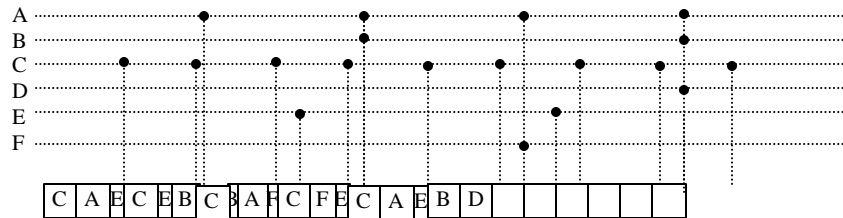


Earliest Deadline First

- At any given time, task with the earliest deadline goes
 - Obviously, very high overhead
 - Have to keep checking deadline
- Dynamic Priority
- Feasible if processor usage < 1
- For task running 5ms (usage .9585)
 - Assume deadline=period

Process	Period	Priority
A	25 ms	5
B	50 ms	3
C	12 ms	6
D	100 ms	1
E	40 ms	4
F	75 ms	2

Deadlines



39