CS 140 Machine Problem 1

CS 140 Machine Problem 1

Deadline: 2 months

Feb. 14, 2014 (you can pass earlier)

Objective

- To design and implement an algorithm for synchronization and concurrent execution of processes

“Welcome to Kitchen Stadium…”

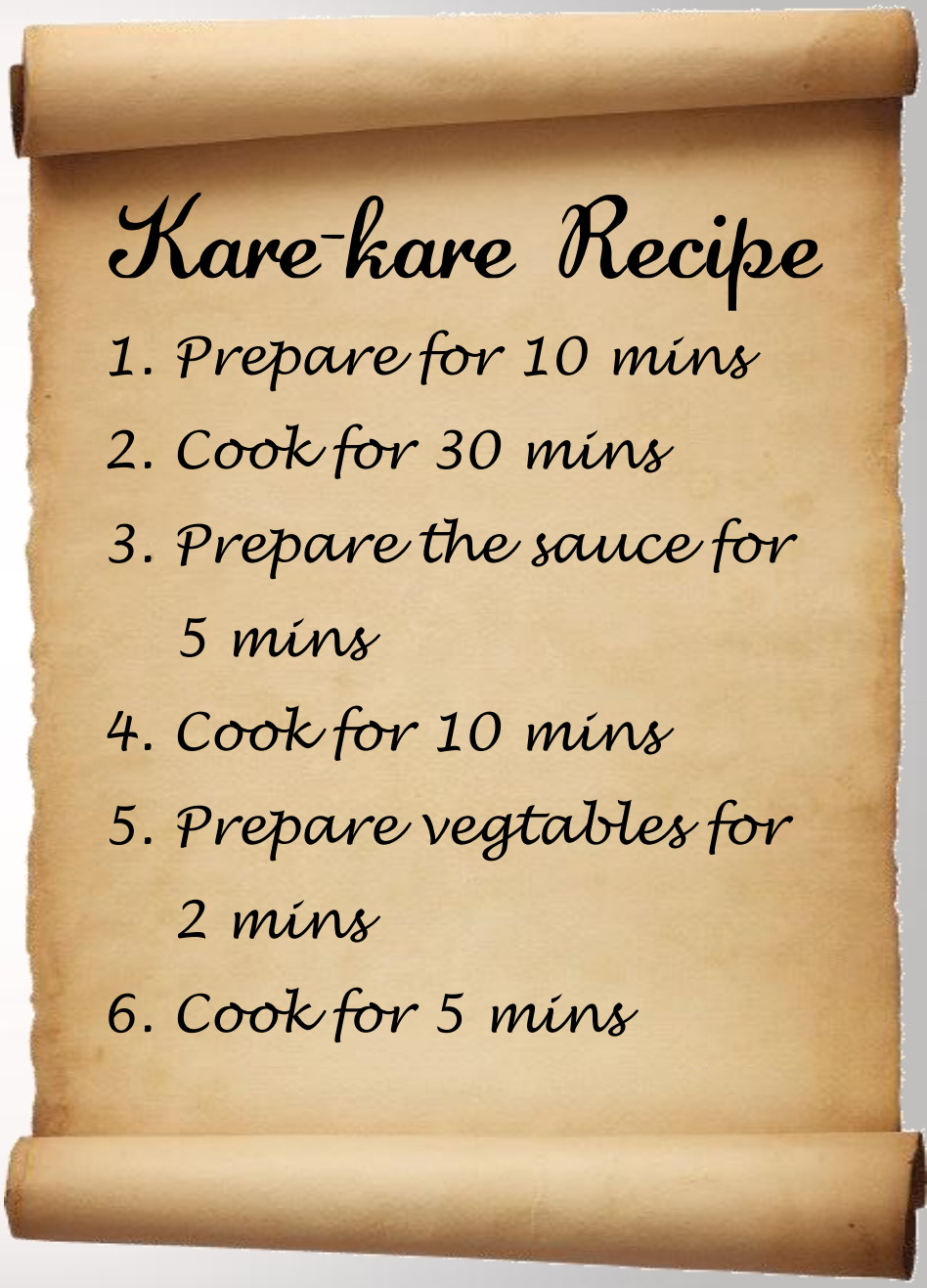
- In this season, you will be tested according to your multi-tasking skills in the kitchen.

The Challenge

- You will be given **a set of dishes** to be prepared.
- Each dish will be ordered at a **specific time in the future**.

The Recipe

- The recipe will contain the instructions on how to cook a particular dish and the duration for each step.
- Each step can be a
  - cooking step (CPU burst)
  - preparation step (IO burst).



The Assistants

You will be having a set of assistants to help you prepare the dish.

With your assistants, you can **prepare** multiple dishes **simultaneously**.

The Problem

For this challenge, you are only given a single stove. Only a **single dish** can be cooked at any given time.

Your Task

- Your task is to design a scheduler to determine which dish should be cooked at any given time.
- To help you determine what to be cooked next at the stove, you can check the details of the following
  - dishes waiting to on the stove
  - dish being cooked

Your Task

- When changing from one dish to another, leave the stove empty for 1 min to clean the pan (CS = 1min)
  - If the stove is already empty, you can use the stove immediately
- Assistants can work while you clean the pan (during CS)

Design Considerations

- Total amount of time to finish all the dishes
- Stove utilization
- Waiting time of the dishes

Sample Schedule

Time	Stove	Ready	Assistants	Remarks
1	adobo(cook=5)	none	none	adobo arrives
2	adobo(cook=4)	none	none	none
3	adobo(cook=3)	soup(cook=5)	none	soup arrives
4	adobo(cook=2)	soup (cook=5)	karekare(prep=5)	karekare arrives
5	adobo(cook=1)	soup (cook=5)	karekare(prep=4) tinola(prep=3)	tinola arrives
6	empty	soup (cook=5)	karekare(prep=3) tinola(prep=2) adobo(prep=6)	adobo cook done Still cleaning the pan… Next is soup
7	soup(cook=5)	none	karekare(prep=2) tinola(prep=1) adobo(prep=5)	none
8	soup(cook=4)	tinola(cook=9)	karekare(prep=1) adobo(prep=4)	tinola prep done

Technical Specifications

Programming Language

- Since this is not a programming subject, you are free to use the following languages as long as your program accepts the input file and produces the output file:
  - C
  - C#
  - C++
  - Java
  - Python
- External libraries are allowed as long as you know what you are doing

Scheduler

- At every minute, the scheduler will determine which dish is to be cooked at that time
- To implement the scheduler, you need to use a function that will take the contents of the stove and the dishes as parameters and will return the dish to be cooked.

```
Dish whatIsNext(Dish dishAtStove, List waitingDishes)
```

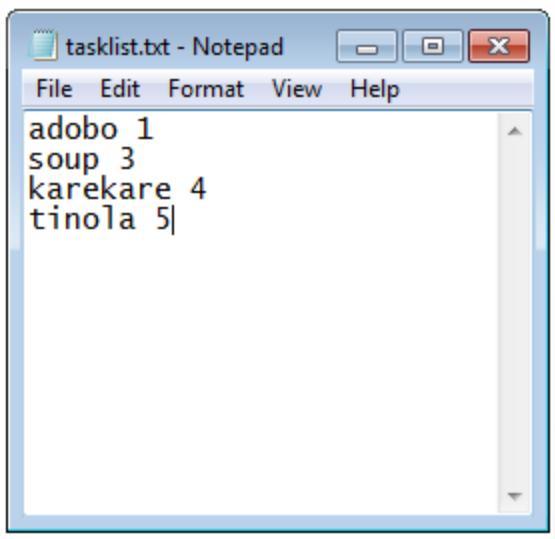
Assumptions and Constraints

- The stove is initially empty
- You can use multiple queues in your design

PROGRAM INPUT

The Task List File

- The cooking task list file (stored as **tasklist.txt**) indicates the order at which dish to start cooking. The following is a sample tasklist file



Each line of tasklist file consists of a single word indicating what dish to start cooking at that particular instant of time. This task file says that at adobo will be arrive at time 1, tinola at time 5 etc. When a dish order arrives, your program loads the appropriate recipe file. Adobo.txt for adobo, tinola.txt for tinola, etc.

Assumptions and Constraints

- Task list
  - A dish can appear only once
  - You can have at least 1 dish in the task list
  - You cannot know the dishes that will come in advance
    - You will not know that karekare will arrive at time=4 during time=1

The Recipe Files

- A recipe file is a text file that contains instructions on how to cook a particular dish. The following is a sample recipe file.

- The first line is the name of the dish followed by its **priority** (1-10, 10 highest)
- Each line of the recipe file contains one word that describes and **action** followed by an **integer** indicating how long the action takes. In our example, prep 10 means karekare will spend 10 mins on the preparation followed by 30 mins of cooking, etc…

Assumptions and Constraints

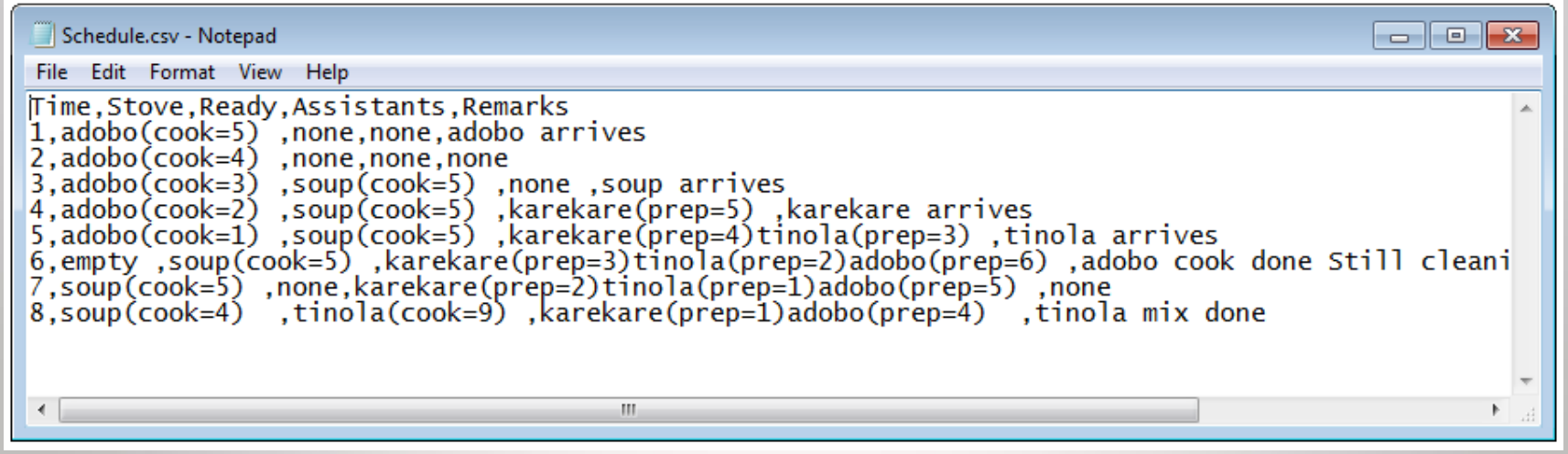
- Recipes
  - Each recipe must contain at least 1 step
  - Each step has a non zero duration
  - The first step can be a cook step or a preparation step
  - Each recipe alternates from preparation step and cooking step
  - Once the recipe is loaded, you can examine the duration of all the steps

PROGRAM OUTPUT

Output File

- The output file must contain the process execution table in CSV (comma separated values) format

Time	Stove	Ready	Assistants	Remarks
1	adobo(cook=5)	none	none	adobo arrives
2	adobo(cook=4)	none	none	none
3	adobo(cook=3)	soup(cook=5)	none	soup arrives
4	adobo(cook=2)	soup (cook=5)	karekare(prep=5)	karekare arrives
5	adobo(cook=1)	soup (cook=5)	karekare(prep=4) tinola(prep=3)	tinola arrives
6	empty	soup (cook=5)	karekare(prep=3) tinola(prep=2) adobo(prep=6)	adobo cook done Still cleaning the pan… Next is soup
7	soup(cook=5)	none	karekare(prep=2) tinola(prep=1) adobo(prep=5)	none
8	soup(cook=4)	tinola(cook=9)	karekare(prep=1) adobo(prep=4)	tinola mix done



Output File

- The output file must contain the process execution table in CSV (comma separated values) format
- You also include the performance metrics of your design
  - Total amount of time to finish all the dishes
  - Stove utilization
  - Weighted average waiting time of all the dishes
    - (∑ waiting time x priority) / (total weight)

DELIVERABLES

1. Source code of the program
2. Documentation
3. Team evaluation
   - Maximum of 4 members

Documentation

Group Member Names

1. Introduction
    (a) Goals and objectives (this design is best for…)
    (b) Major constraints
2. Algorithm Design
    (a) Data structures used
    (b) Pseudocode on how the selection is made
    (c) Algorithm Performance
        1. Best-case performance
        2. Worst-case performance
        3. Average-case performance

Team Evaluation

Indicate the following

- Group mate names
- Your major contributions
- Division of work
- Evaluation

Grading

- Originality of algorithm design 30%
- Documentation (design, code) 15%
- Correctness of implementation 45%
  - (no bugs from your part of the code =))
- Teamwork 10%

Submissions

- Deadline: Feb 14, 2014 (11:59 pm)
- Email:ISubmitMyCode@gmail.com
- Subject: CS 140 MP
- Contents:
  - List of members
  - Source code
  - Documentation
  - Team evaluation (in a separate email)

Final Notes

- This is meant as a group exercise. You may discuss the MP with other groups, but you are expected to come up with your own **original solution** to the problem.
- A **demo** of your implementation will be scheduled.
  - During this demo all members of the group should be present and be ready to answer questions regarding the design of the algorithm and its implementation.
- Points will be deducted for late submissions
  - (5% for every working day late, i.e., not including Saturdays, Sundays, holidays, and class suspensions).
- Submission of the MP (even if it will get a grade of zero due to deductions) is strictly required for the course.