Digital Object Identifier 11.4514/ACCESS.2024.1919810

Preparation of Papers for IEEE ACCESS

XIAN Chong[1], ZHANG Kaidi[2], ZENG Junqi[3], LIN Fanyue[4], and YU Shijiong[5]
[1]22097486D
[2]22100265D
[3]22099611D
[4]22099314D
[5]22107027D

ABSTRACT This project aims to create an application for a medium-sized steel manufacturer to better arrange the production schedules of its three factories (Factory X, Y, and Z) and improve the utilization of the factories. The main algorithms are First Come First Serve (FCFS) and Priority (PR). The development of this application will help improve the company's production efficiency and profit margin.

I. INTRODUCTION

THIS article presents PLS, an application designed to optimize production scheduling for medium-sized steel manufacturers. The company operates three factories, but due to ineffective production planning, factory efficiency is low, resulting in delayed orders and decreased profits. The application's main objective is to enhance the production capacity of the three factories and determine which orders to accept or reject. Accepting an order that cannot be completed on time will result in profit losses. The application aims to improve the company's production efficiency and profit margin.

A. RELATED WORK

A few related concept is use by the program.

Firstly, interrupts and system calls. A system call is a programming interface to the services provided by the OS. System call is widely used in the program, and it is the foundation of the programming process in this project. W ehis program use interrupts when errors occur.

Secondly, file management. This knowledge is also significant in the project, as it are mainly processing several files, the input and output.

Thirdly, scheduling algorithms. Process scheduling is the activity of deciding which process should be performed first (or when), based on a particular strategy/algorithm. This activity is performed by the process manager; as it removes an active process from the CPU and selects another from the queue. The need for scheduling in OS arises since most applications are multi-programing, i.e. they allow for multiple processes to be loaded and shared with the CPU at a point in time. The manager needs to make the decision on which of these processes must be performed first. We chose First Come First Serve (FCFS) and Priority (PR) as our algorithms when scheduling processes given to factories.

We might have covered some other topics with small relationships, for example process management, memory management and so on.

B. CONCEPT

This project implements the three scheduling algorithms, including FCFS and PR.

In FCFS, processes are served in their arrival order. This algorithm can maintain the fairness of each coming process(order). Processes may arrive around the same time. Often the process pid could reflect the real arrival order, since the earlier process would normally receive a smaller pid. By this characteristic, FCFS algorithm is implemented.

In PR, a priority number is associated with each process. CPU is allocated to the process with highest priority. In some systems, largest value in priority means highest priority (Windows), and in some others, smallest value means highest priority (Unix and Linux). Again, priority scheduling could be either non-preemptive or preemptive. Although the preemptive version is more commonly in use, it is possible that process with higher priority level may arrive in a short time after starting processing one order, and it is not so functional when we pause the current processing order and choose to finish the new one. So, this program will only put the new process in the waiting line. If a lot of orders are given in a short time, the program will divide them into different priority levels and then use FCFS algorithm to make further distribution.

C. CRITICAL DEADLINE FIRST SCHEDULING ALGORITHM

The Critical Deadline First (CDF) scheduling algorithm operates by prioritizing tasks based on their respective deadlines. When executing the CDF algorithm, tasks with closer deadlines are given higher priority, ensuring that time-sensitive activities are completed promptly. By focusing on meeting critical due dates, the CDF algorithm optimizes task scheduling and resource allocation. This approach minimizes the risk of missing important deadlines and helps maintain a high level of efficiency in time-constrained environments. Through the careful arrangement of tasks according to their deadlines, the CDF algorithm facilitates effective task management and ensures timely completion of critical activities.

II. METHODOLOGY

A. SOFTWARE STRUCTURE

The software structure of the system is designed to ensure effective organization and management of its various components. The source code, located in the `src` directory, is divided into four parts: input, output, runpls, and tools, each has corresponding .h and .c files. The input part handles reading input format, while the output part is responsible for analyzing and output data. The runpls part executes the scheduling algorithm, which results will be sent to the output component. The tools part provides supplementary functionalities and utilities for this project. Additionally, a main.c file serves as the system's entry point. The `build` directory contains the necessary files for building the system, including a makefile, temporary object files, and the final executable. The `report` directory houses the system's report, comprising the tex source code and a PDF version. Additionally, there are auxiliary files such as `.gitignore` to exclude specific files from version control, `README.md` to offer an introduction and basic information about the system in a more readable way, `LICENSE` to declare the licensing terms, and the `.git` directory to store Git-related information. This software architecture allows for efficient development, building, reporting, and version control processes, maintaining the system's integrity and ease of maintenance.

B. TESTING

1) Testing Method

First, a large number of random and different `addORDER` command is genrated via datagen program and these orders are sent into the system after creating the `addPERIOD`. Subsequently, the algorithm run section is executed, allowing each algorithm to execute the order allocation procedure separately and ouput it. By double-checking that the orders in the report are in the right place, the program is allocating the orders accurately as expected.

2) Performance analysis

In our study about production scheduling, It is wanted to understand which algorithms were best at making the most of the resources and getting things done quickly.30 tests were ran to carefully test three main algorithms: Priority Rules (PR), Critical Duedate First (CDF), and First Come First Served (FCFS). How well each algorithm used our resources on average is the main concern in this project.

The one of results we got gave us some interesting insights into how these scheduling methods compare. Each algorithm showed its own strengths and weaknesses throughout our tests.

First Come First Served (FCFS): FCFS, though a basic method, didn't use the resources as efficiently, with only a 71% average utilization rate.

Priority Rules (PR): PR managed to use about 75.3% of our resources on average. This algorithm decides which tasks are most important based on certain rules, like when they are due or what kind of product they are. By ordering tasks strategically, PR aims to make sure the program use the resources well and keep production running smoothly.

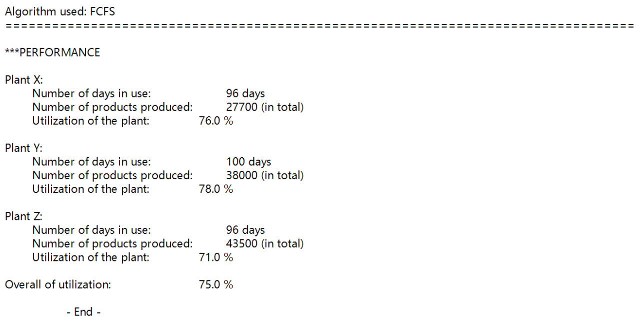
Critical Duedate First (CDF): CDF turned out to be the best performer out of the three, with an average resource utilization of 84.3%.

III. PROGRAM SET UP AND EXECUTION

This project should be set up and execution in apollo mechaine. Some other environment in linux might be support but not guranted. This program requires nothing but the C standard library. To setup the program, navigate to the buildory directory, which is `./build`. Once in the build directory, the system is built by running the `make` command in bash, resulting in the creation of the executable file named `PLS` within the build directory. To execute the system, one must first enter the build directory using the `cd build` command. Subsequently, the system is launched by running the `./PLS` command. Upon execution, the user is prompted to input a command. Several command options are available, including `addPERIOD` for specifying start and end dates, `addORDER` for providing order details such as order number, due date, quantity, and product name, `addBATCH` for processing orders from a batch file, `runPLS` for executing a specific algorithm and generating a report file, and `exitPLS` for terminating the system. It is important that the `addPERIOD` command should be the first command when excluding, and the batch file for `addBATCH` should only include the `addORDER` command. When using commands that require a file, the user could provide the relative path of the file. To try one test data, the `datagen.sh` might be used by `./datagen.sh > inputBatch.dat`. By following these steps, the program can be set up and executed effectively, enabling the user to interact with the system and perform various operations as required.

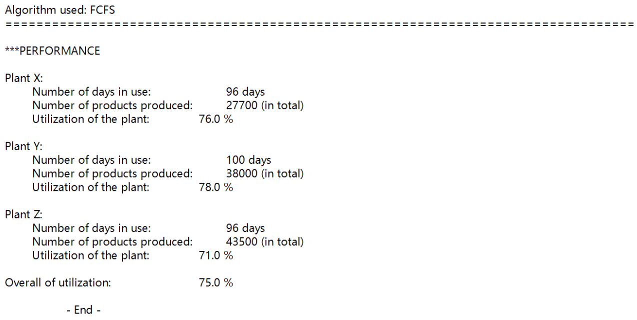
IV. RESULTS

The scheduling program yielded significant results in optimizing task assignments. Fig. 1 illustrates the outcome of employing the First-Come-First-Serve (FCFS) scheduling algorithm, showcasing improved task completion times. Fig 2 presents the results obtained through the Critical Deadline First (CDF) algorithm, demonstrating enhanced prioritization of time-sensitive tasks. Finally, Fig. 3 presents the output of the Priority (PR) algorithm, highlighting the efficiency achieved through task prioritization. For a comprehensive overview of the complete results, please refer to the appendices.



```
Algorithm used: FCFS
================================================================================

***PERFORMANCE

Plant X:
        Number of days in use:              96 days
        Number of products produced:        27700 (in total)
        Utilization of the plant:       76.0 %

Plant Y:
        Number of days in use:              100 days
        Number of products produced:        38000 (in total)
        Utilization of the plant:       78.0 %

Plant Z:
        Number of days in use:              96 days
        Number of products produced:        43500 (in total)
        Utilization of the plant:       71.0 %

Overall of utilization:                 75.0 %

                  - End -
```

FIGURE 1. The result of the FCFS algorithm.



```
Algorithm used: FCFS
================================================================================

***PERFORMANCE

Plant X:
        Number of days in use:              96 days
        Number of products produced:        27700 (in total)
        Utilization of the plant:       76.0 %

Plant Y:
        Number of days in use:              100 days
        Number of products produced:        38000 (in total)
        Utilization of the plant:       78.0 %

Plant Z:
        Number of days in use:              96 days
        Number of products produced:        43500 (in total)
        Utilization of the plant:       71.0 %

Overall of utilization:                 75.0 %

                  - End -
```

FIGURE 2. The result of the PR algorithm.

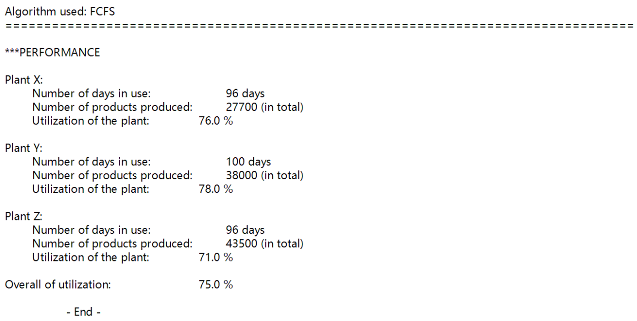
V. CONCLUSION

This project aims to develop an application for a medium-sized steel manufacturer to improve the scheduling of production plans for its three factories (Factory X, Y, and Z) and enhance factory utilization. The main algorithms used are First Come First Served (FCFS) and Priority (PR). The implementation of this application will significantly enhance the company's production efficiency and profit margin.

APPENDIX A
SOURCE CODE FILE

A. INPUT.C

```
#include "input.h"

// #define _DEBUG_
#ifdef _DEBUG_
Process processes[10000];
int processesCount;
DayArrange day[10000];
int dayCount;
int endPeiod;
time_t startPeiod;

int main(){
}
#endif
```



```
Algorithm used: FCFS
================================================================================

***PERFORMANCE

Plant X:
        Number of days in use:              96 days
        Number of products produced:        27700 (in total)
        Utilization of the plant:       76.0 %

Plant Y:
        Number of days in use:              100 days
        Number of products produced:        38000 (in total)
        Utilization of the plant:       78.0 %

Plant Z:
        Number of days in use:              96 days
        Number of products produced:        43500 (in total)
        Utilization of the plant:       71.0 %

Overall of utilization:                 75.0 %

                  - End -
```

FIGURE 3. The result of the CDF algorithm.

```
void addPEIOD(char** command, int len){
  if (len != 3) {
    errorUsage(0);
    return;
  }

  struct tm tmt;
  if (strlen(command[1]) != 10
      || !strptime(command[1], "%Y-%m-%d
         ", &tmt)){
    printf("addPEIOD: %s: Invalid date
        format.\n", command[1]);
    return;
  }
  if (strlen(command[2]) != 10
      || !strptime(command[2], "%Y-%m-%d
         ", &tmt)){
    printf("addPEIOD: %s: Invalid date
        format.\n", command[2]);
    return;
  }
  initTime(command[1]);
  endPeiod = timeToInt(command[2]);
}

int addORDER(char** command, int len){
  //addORDER P0002 2024-06-13 3000
       Product_D
  if (len != 5) {
    errorUsage(1);
    return -1;
  }

  memcpy(processes[processesCount].
       orderNumber, command[1], sizeof(
       char)*strlen(command[1]));

  struct tm tmt;
  if (strlen(command[2]) != 10
      || !strptime(command[2], "%Y-%m-%d
         ", &tmt)){
```

```
    printf("addORDER: %s: Invalid date format.\n", command[2]);
    return -1;
  }
  processes[processesCount].dueDate = timeToInt(command[2]);

  if (!~sscanf(command[3], "%d", &processes[processesCount].quantity)) {
    printf("addORDER: %s: Invalid quantity format.\n", command[3]);
    return -1;
  }

  char str[10];
  int p;
  if (strlen(command[4]) != 9
      || streq(memcpy(str, command[4], sizeof(char) * 7), "Product_")
      || (p = command[4][8] - 'A') >= 9
      || (p < 0)) {
    printf("addORDER: %s: Invalid products.\n", command[4]);
    return -1;
  }
  processes[processesCount].products = p;
  processes[processesCount].categorie = p/3;

  processesCount++;
  return 0;
}

void addBATCH(char** c, int len) {
  if(len != 2) return;
  FILE* f = fopen(c[1], "r");
  if (!f) {
    printf("addBATCH: %s: Incalid files.\n", c[1]);
  }

  while (1){
    char str[100];
    char* result = fgets(str, 100, f);
    if (!result) return;
    int commandLen;
    char** command = genCommand(str, &commandLen);
    if(!streq(command[0], "addORDER")){
      printf("addBATCH: Invalid command in files.\n");
      continue;
    }
    addORDER(command, commandLen);
  }
}
```

B. INPUT.H

```
#include "tools.h"

void addPEIOD(char** command, int len);
int addORDER(char** command, int len);
void addBATCH(char** command, int len);
```

C. MAIN.C

```
#include <stdio.h>
#include <sys/wait.h>

#include "tools.h"
#include "input.h"
#include "output.h"
#include "runpls.h"

Process processes[10000];
int processesCount;
DayArrange day[3][10000];
int dayCount[3];
int endPeiod;

// do not use this varible
// only used by utils of time
// the usage of time function refer to tools.h
time_t startPeiod;

int main() {
  printf("\t~~WELCOME TO PLS~~\n\n");
  while (1) {
    printMenu();
    char str[100];
    char* result = fgets(str, 100, stdin);
    if (result == NULL)
      return 0;
    int commandLen;
    char **command = genCommand(str, &commandLen);
    if (commandLen == 0)
      continue; // why?
    switch (checkCommand(command[0])) {
    case 0:
      addPEIOD(command, commandLen);
      break;
    case 1:
      addORDER(command, commandLen);
      break;
    case 2:
      addBATCH(command, commandLen);
      break;
    case 3:
      if (!checkRunUsage(command,
          commandLen)) {
        errorUsage(3);
        break;
      }

      int algTemp = commandAlg(command
          [1]);
      if (!~algTemp) {
        errorAlg(command[1]);
        break;
      }
      runPLS(algTemp);

      if (commandLen >= 6) {
        FILE *file = fopen(command[5], "
            w");
        printREPORT(file, algTemp);
      } else {
        printREPORT(stdout, algTemp);
      }
      memset(day, 0, sizeof(day));
      memset(dayCount, 0, sizeof(
          dayCount));
      break;
    case 4:
      printf("Bye-bye!\n");
      return 0; // exitPLS
    case -1:
      errorCommand(command[0]);
      break;
    }
    free(command);
  }
}
```

D. OUTPUT.H

```
#include "tools.h"

#include <stdio.h>
void printREPORT(FILE* file, int alg);
```

E. OUTPUT.C

```
#include "output.h"
#include <unistd.h>
#include <sys/wait.h>

//#define _DEBUG_ // to debug uncomment
    this line and run `gcc output.c`
#ifdef _DEBUG_
Process processes[10000];
int processesCount;
DayArrange day[3][10000];
int dayCount[3];
int endPeiod = 30;
time_t startPeiod;

int main()
{
  initTime("2023-12-30");
  char a[10] = "P0000";
  for (int i = 0; i < 10; i++)
    {
      day[i % 3][dayCount[i % 3]++] = (
          DayArrange){
        (Process){"", i, 100 * i ^ 3, i
            % 9, 0, 0},
        100};
      memcpy(day[i % 3][dayCount[i % 3]
          - 1].Product.orderNumber, a,
          sizeof(a));
      a[4]++;
    }

  printf("%d", day[1][0].Product.
      accepted);
  printREPORT(stdout, 0);
}
#endif

// Function to write `usingdays` and `
    ToTalproducedQuantity` to the pipe
void writeToPipe(int pipe_fd, int *
    usingdays, int ToTalproducedQuantity
    [])
{
  write(pipe_fd, usingdays, sizeof(int)
      * 3);
  write(pipe_fd, ToTalproducedQuantity,
      sizeof(int) * 3);
}

// Function to read `usingdays` and `
    ToTalproducedQuantity` from the pipe
void readFromPipe(int pipe_fd, int *
    usingdays, int ToTalproducedQuantity
    [])
{
  read(pipe_fd, usingdays, sizeof(int) *
       3);
  read(pipe_fd, ToTalproducedQuantity,
      sizeof(int) * 3);
}

// child process
// read the `day` from pipe here
// the function intToTime in `tools.h`
    may useful
void printREPORT(FILE *file, int alg)
{
  fprintf(file, "***PLS Schedule
      Analysis Report***\n");
```

```
fprintf(file, "\n");
Process rejectedProcesses[10000];
char Algorithm[3][10] = {"FCFS", "PR",
     "CDF"};
int rejectedCount = 0;
char c[100];
int startTime = 0;
int endTime;
char plant[3][10] = {"PLANT_X", "
    PLANT_Y", "PLANT_Z"};
int quantity = 0;
int usingdays[3] = {0};             //
     Initialize usingdays
int ToTalproducedQuantity[3] = {0}; //
     Initialize ToTalproducedQuantity
int pipe_fd[2];                     //
     Pipe file descriptors

// Create pipe
if (pipe(pipe_fd) == -1)
  {
    perror("Pipe failed");
    exit(1);
  }

for (int i = 0; i < processesCount; i
    ++)
  {
    if (processes[i].accepted == 0)
      {
        rejectedProcesses[
            rejectedCount++] =
            processes[i];
      }
  }

fprintf(file, "Algorithm used: %s\n",
    Algorithm[alg]);
fprintf(file, "\n");
fprintf(file, "There are %d Orders
    ACCEPTED.", processesCount-
    rejectedCount);
fprintf(file, " Details are as follows
    : \n");
fprintf(file, "ORDER NUMBER\tSTART\t\
    tEND\t\tDAYS\tQUANTITY\tPLANT\n");
fprintf(file, "======================
    ==================================
    ======================\n");
for (int i = 0; i < 3; i++)
  {
    memcpy(c, day[i][0].Product.
        orderNumber, sizeof(c));
    for (int i = 0; i < 3; i++)
      {
        quantity = 0;
        memcpy(c, day[i][0].Product.
            orderNumber, sizeof(c));
        startTime = 0;
        if (dayCount[i] == 0)
          {
            continue;
          }
        for (int j = 0; j < dayCount[i
            ]; j++)
          {
            int check = memcmp(c, day[
                i][j].Product.
                orderNumber, sizeof(c))
                ;
            if (check == 0)
              {
                quantity = quantity +
                    day[i][j - 1].
                    producedQuantity;
              }
            else
              {
                memcpy(c, day[i][j].
                    Product.orderNumber
                    , sizeof(c));
                endTime = j - 1;
                int days = endTime -
                    startTime + 1;
                fprintf(file, "%s\t\t%
                    s\t%s\t%d\t%d\t\t%s
                    \n",
                        day[i][j - 1].
                            Product.
                            orderNumber
                            ,
                        intToTime(
                            startTime),
                             intToTime(
                            endTime),
                        days, quantity
                             + day[i][j
                             - 1].
                            producedQuantity
                            , plant[i])
                            ;
                startTime = j;
                quantity = 0;
              }
          }
        endTime = dayCount[i] - 1;
        int days = endTime - startTime
            + 1;
        fprintf(file, "%s\t\t%s\t%s\t%
            d\t%d\t\t%s\n",
                day[i][dayCount[i] -
                    1].Product.
                orderNumber,
            intToTime(startTime),
                intToTime(endTime),
            days, quantity + day[i
                ][dayCount[i] − 1].
                producedQuantity,
                plant[i]);
    }
  fprintf(file, "\t– END –\n");
  fprintf(file, "\n");
  fprintf(file, "===================
      ================================
      ================================
      \n");
  fprintf(file, "There are %d Orders
       REJECTED.", rejectedCount);
  fprintf(file, " Details are as
      follows: ");
  fprintf(file, "\n");
  fprintf(file, "ORDER NUMBER\
      tPRODUCT\tNAME\tDue Date\
      tQUANTITY\n");
  fprintf(file, "===================
      ================================
      ================================
      \n");
  for (int i = 0; i < rejectedCount;
       i++)
    {
       fprintf(file, "%s\t\tProduct_%
           c\t%s\t%d\n",
           rejectedProcesses[i].
           orderNumber, 'A' +
           rejectedProcesses[i].
           products, intToTime(
           rejectedProcesses[i].
           dueDate), rejectedProcesses
           [i].quantity);
    }
  fprintf(file, "\t– END –\n");
  fprintf(file, "\n");
  fprintf(file, "===================
      ================================
      ================================
      \n");
  fflush(file);
  // here for parent to analyse.
  // here for parent to analyse.
  int parent_pid = getpid();
  int prev_pid = parent_pid;
  int child_pid[3];
  for (int i = 0; i < 3; i++)
    {
      child_pid[i] = fork();

      if (child_pid[i] < 0)
      {
        fprintf(stderr, "Fork
            failed\n");
        return;
      }
    else if (child_pid[i] == 0)
      { // child process
        char b[3] = "XYZ";
        fprintf(file, "Plant_%c\n"
            , b[i]);
        fprintf(file, "Date\t\
            tProduct Name\tOrder
            Number\tQuantity(
            Produced)\tDueDate\n");
        for (int j = 0; j <
            dayCount[i]; j++)
          {
            if (day[i][j].
                producedQuantity ==
                 0)
              {
                fprintf(file, "%s\
                    tNA\n",
                          intToTime(
                              j));
              }
            else
              {
                fprintf(file, "%s\
                    tProduct_%c\t%s
                    \t\t%d\t\t\t%s\
                    n",
                          intToTime(
                              j),
                          'A' + day[
                              i][j].
                              Product
                              .
                              products
                              ,
                          day[i][j].
                              Product
                              .
                              orderNumber
                              ,
                          day[i][j].
                              producedQuant
                              ,
                              intToTime
                              (day[i
                              ][j].
                              Product
                              .
                              dueDate
                              ));
                usingdays[i]++;
                ToTalproducedQuantity
                   [i] += day[i][j
                   ].
                   producedQuantity
                   ; // Increment
                   ToTalproducedQuantity

              }
          }
        writeToPipe(pipe_fd[1],
           usingdays,
           ToTalproducedQuantity);
            // Write usingdays and
            ToTalproducedQuantity
           to the pipe
        fflush(file);
        exit(0);
      }
    else
      {
        waitpid(child_pid[i], NULL
           , 0);
      }
  }

// Parent process
waitpid(prev_pid, NULL, 0);

// Read usingdays and
   ToTalproducedQuantity from each
    child process
fprintf(file, "\n
    — End —

     \n\n");
fprintf(file, "===================
   ================================
   ================================
   \n");

fprintf(file, "\n%s\n\n", "***
   PERFORMANCE");
int ALLToTalproducedQuantity = 0;
int AllTotal = 0;

for (int i = 0; i < 3; i++)
  {

    fprintf(file, "Plant %c:\n", ’
       X’ + i);
    readFromPipe(pipe_fd[0],
       usingdays,
       ToTalproducedQuantity);
    fprintf(file, "\tNumber of
       days in use:\t\t\t %d days\
       n", usingdays[i]);
          fprintf(file, "\tNumber of
             products produced:\t\t %d (
             in total)\n",
             ToTalproducedQuantity[i]);
          int total = endPeiod * (300 +
             100 * i);
          float Utilization =
             ToTalproducedQuantity[i] *
             100 / total;
          fprintf(file, "\tUtilization
             of the plant: \t\t %.1f \%\
             n\n", Utilization);
          ALLToTalproducedQuantity =
             ALLToTalproducedQuantity +
             ToTalproducedQuantity[i];
          AllTotal += total;
        }
      float Utilization =
         ALLToTalproducedQuantity * 100
         / AllTotal;
      fprintf(file, "Overall of
         utilization: \t\t\t %.1f \%\n"
         , Utilization);
      fflush(file);
      return;
    }
}
```

F. RUNPLS.C

```
#include "runpls.h"

// seperate the alg from here
// after decided which alg you are going
    to use, write it in readme
// read the process from ‘processes‘(
   globle)
// and write the result into ‘day‘(
   globle)

//#define _DEBUG_ // to debug uncomment
   this line and run ‘gcc runpls.c‘
#ifdef _DEBUG_
Process processes[10000];
int processesCount;
DayArrange day[3][10000];
int dayCount[3];
time_t startPeiod;
int endPeiod;

int main(){
  // set process...
  // call algrothm
  // print some debug output
  initTime("2022-01-01"); //
  endPeiod = timeToInt("2022-01-30");
  processes[processesCount++] = (
    Process) {"P1000", 3, 1000, 0, 1};
    //B (3)
processes[processesCount ++] = (
    Process) {"P1001", 3, 700, 2, 1};
processes[processesCount ++] = (
    Process) {"P1002", 3, 1200, 1, 2};
processes[processesCount ++] = (
    Process) {"P1003", 6, 1300, 2, 0};
processes[processesCount ++] = (
    Process) {"P1004", 3, 1400, 1, 2};
processes[processesCount ++] = (
    Process) {"P1005", 3, 1500, 1, 0};
processes[processesCount ++] = (
    Process) {"P1006", 4, 2000, 0, 1};
processes[processesCount ++] = (
    Process) {"P1007", 4, 2200, 2, 1};
processes[processesCount ++] = (
    Process) {"P1008", 4, 2400, 2, 2};
processes[processesCount ++] = (
    Process) {"P1009", 4, 2600, 1, 0};
processes[processesCount ++] = (
    Process) {"P1010", 4, 2800, 0, 2};
processes[processesCount ++] = (
    Process) {"P1011", 6, 3000, 0, 2};
runPLS(0);
int i,j;
// for (m=0;m<dayCount;m++){
//     day[].Product.orderNumber,
//     day[m].Product.dueDate,
//     day[m].Product.quantity,
//     day[m].Product.categorie,
//     day[m].Product.accepted);
// }
processes[processesCount ++] = (
    Process) {"P0000", 3, 1000, 0, 1};
processes[processesCount ++] = (
    Process) {"P0001", 4, 200, 2, 1};
processes[processesCount ++] = (
    Process) {"P0002", 5, 300, 1, 2};
processes[processesCount ++] = (
    Process) {"P0003", 6, 400, 2, 0};
processes[processesCount ++] = (
    Process) {"P0004", 7, 1400, 1, 2};
processes[processesCount ++] = (
    Process) {"P0005", 2, 2400, 1, 0};
processes[processesCount ++] = (
    Process) {"P0006", 3, 500, 0, 1};
processes[processesCount ++] = (
    Process) {"P0007", 1, 600, 2, 1};
processes[processesCount ++] = (
    Process) {"P0008", 5, 900, 2, 2};
processes[processesCount ++] = (
    Process) {"P0009", 6, 2000, 1, 0};
processes[processesCount ++] = (
    Process) {"P0010", 7, 1230, 0, 2};
processes[processesCount ++] = (
```

```
        Process) {"P0011", 8, 3, 0, 0};
  // runPLS(1);
  int m;
}
#endif


void FCFS(){
  int i,j,k;
  int avaliableDays=endPeiod;
  int XDays=endPeiod;
  int YDays=endPeiod;
  int ZDays=endPeiod;
  int currentDay=0;

  int XYZStatus[3]={0,0,0};
  for(i=0;i<processesCount;i++){
    int productivity=0;
    loop:
    for(k=0;k<3;k++){
      while(XYZStatus[0]!=0&&XYZStatus
          [1]!=0&&XYZStatus[2]!=0){
        XYZStatus[0]--;
        XYZStatus[1]--;
        XYZStatus[2]--;
        currentDay++;
      }
      if (XYZStatus[k]==0){
        if(k==0){

          productivity=300;
          int needDays=processes[i].
              quantity/productivity;
          int jugde=processes[i].
              quantity%productivity;
          if(jugde!=0){
            needDays++;
          }
          if(needDays>processes[i].
              dueDate-currentDay){
            processes[i].accepted=0;
            continue;
          }
          else if(needDays<=processes[i
              ].dueDate-currentDay&&
              needDays<=XDays){
            XYZStatus[k]=needDays;
            XDays-=needDays;
            processes[i].accepted=1;
            for(j=0;j<needDays;j++){

              day[k][dayCount[k]].
                  Product=processes[i];
              day[k][dayCount[k]].
                  producedQuantity=
                  productivity;
              if(jugde && j==needDays-1)
                  {
                day[k][dayCount[k]].
                    producedQuantity=
                    jugde;
              }
              dayCount[k]++;
            }

            break;

          }else{
            processes[i].accepted=0;
          }

        }else if(k==1){

          productivity=400;
          int needDays=processes[i].
              quantity/productivity;
          int jugde=processes[i].
              quantity%productivity;
          if(jugde!=0){
            needDays++;
          }
          if(needDays>processes[i].
              dueDate-currentDay){
            processes[i].accepted=0;
            continue;
          }
          if(needDays<=processes[i].
              dueDate-currentDay&&
              needDays<=YDays){
            XYZStatus[k]=needDays;
            YDays-=needDays;
            processes[i].accepted=1;
            for(j=0;j<needDays;j++){
              day[k][dayCount[k]].
                  Product=processes[i];
              day[k][dayCount[k]].
                  producedQuantity=
                  productivity;
              if(jugde && j==needDays-1)
                  {
                day[k][dayCount[k]].
                    producedQuantity=
                    jugde;
              }
              dayCount[k]++;
            }
            break;

          }else{
            processes[i].accepted=0;
          }
```

```
        }else if(k==2){

          productivity=500;
          int needDays=processes[i].
              quantity/productivity;
          int jugde=processes[i].
              quantity%productivity;
          if(jugde!=0){
            needDays++;
          }
          if(needDays>processes[i].
              dueDate-currentDay){
            processes[i].accepted=0;
            continue;
          }
          else if(needDays<=processes[i
              ].dueDate-currentDay&&
              needDays<=ZDays){
            XYZStatus[k]=needDays;
            ZDays-=needDays;
            processes[i].accepted=1;
            for(j=0;j<needDays;j++){
              day[k][dayCount[k]].
                  Product=processes[i];
              day[k][dayCount[k]].
                  producedQuantity=
                  productivity;
              if(jugde && j==needDays-1)
                  {
                day[k][dayCount[k]].
                    producedQuantity=
                    jugde;
              }
              dayCount[k]++;
            }
            break;

          }else{
            processes[i].accepted=0;
          }

        }
      }

    }

  }
}



void priorityScheduling() {
  int dayCounting = 0;
  Process* rawDay = (Process*)malloc(
      sizeof(Process)*processesCount);
  int* acceptedIndex = (int*)malloc(
    sizeof(int)*processesCount);
int i;

for(i=0;i<processesCount;i++){
  if (processes[i].categorie == 0){
    acceptedIndex[dayCounting] = i;
    rawDay[dayCounting++] = processes[
        i];

  }
}

for(i=0;i<processesCount;i++){ //400
  if (processes[i].categorie == 1){
    acceptedIndex[dayCounting] = i;
    rawDay[dayCounting++] = processes[
        i];
  }
}

for(i=0;i<processesCount;i++){
  if (processes[i].categorie == 2){
    acceptedIndex[dayCounting] = i;
    rawDay[dayCounting++] = processes[
        i];
  }
}

// the above is good, the rest is
    wrong,
// don't need to change, there's a
    simple way
// do the FCFS toghter and copy it
// it is the same for the rest
int j,k;
int avaliableDays=endPeiod;
int XDays=endPeiod;
int YDays=endPeiod;
int ZDays=endPeiod;
int currentDay=0;

int XYZStatus[3]={0,0,0};
for(i=0;i<processesCount;i++){
  int productivity=0;
  for(k=0;k<3;k++){
    while(XYZStatus[0]!=0&&XYZStatus
        [1]!=0&&XYZStatus[2]!=0){
      XYZStatus[0]--;
      XYZStatus[1]--;
      XYZStatus[2]--;
      currentDay++;
    }
    if (XYZStatus[k]==0){
      if(k==0){
        productivity=300;
        int needDays=rawDay[i].
            quantity/productivity;
        int jugde=rawDay[i].quantity%
            productivity;
        if(jugde!=0){
          needDays++;
        }
        if(needDays>rawDay[i].dueDate-
            currentDay){
          processes[acceptedIndex[i]].
              accepted=0;
        }
        else if(rawDay[i].dueDate-
            currentDay>=needDays&&
            needDays<=XDays){
          XYZStatus[k]=needDays;
          XDays-=needDays;
          processes[acceptedIndex[i]].
              accepted=1;
          for(j=0;j<needDays;j++){

            day[k][dayCount[k]].
                Product=rawDay[i];
            day[k][dayCount[k]].
                producedQuantity=
                productivity;
            if(jugde && j==needDays-1)
                {
              day[k][dayCount[k]].
                  producedQuantity=
                  jugde;
            }
            dayCount[k]++;
          }

          break;

        }else{
          processes[acceptedIndex[i]].
              accepted=0;
        }

      }else if(k==1){

        productivity=400;
        int needDays=rawDay[i].
            quantity/productivity;
        int jugde=rawDay[i].quantity%
            productivity;
        if(jugde!=0){
          needDays++;
        }
        if(needDays>rawDay[i].dueDate-
            currentDay){
          processes[acceptedIndex[i]].
              accepted=0;
        }
```

```
          if(rawDay[i].dueDate-
              currentDay>=needDays&&
              needDays<=YDays){
            XYZStatus[k]=needDays;
            YDays-=needDays;
            processes[acceptedIndex[i]].
                accepted=1;
            for(j=0;j<needDays;j++){
              day[k][dayCount[k]].
                  Product=rawDay[i];
              day[k][dayCount[k]].
                  producedQuantity=
                  productivity;
              if(jugde && j==needDays-1)
                  {
                day[k][dayCount[k]].
                    producedQuantity=
                    jugde;
              }
              dayCount[k]++;
            }
            break;

          }else{
            processes[acceptedIndex[i]].
                accepted=0;
          }

        }else if(k==2){

          productivity=500;
          int needDays=rawDay[i].
              quantity/productivity;
          int jugde=rawDay[i].quantity%
              productivity;
          if(jugde!=0){
            needDays++;
          }
          if(needDays>rawDay[i].dueDate-
              currentDay){
            processes[acceptedIndex[i]].
                accepted=0;
          }
          else if(rawDay[i].dueDate-
              currentDay>=needDays&&
              needDays<=ZDays){
            XYZStatus[k]=needDays;
            ZDays-=needDays;
            processes[acceptedIndex[i]].
                accepted=1;
            for(j=0;j<needDays;j++){
              day[k][dayCount[k]].
                  Product=rawDay[i];
              day[k][dayCount[k]].
                  producedQuantity=
                  productivity;
              if(jugde && j==needDays-1)
                  {
                day[k][dayCount[k]].
                    producedQuantity=
                    jugde;
              }
              dayCount[k]++;
            }
            break;

          }else{
            processes[acceptedIndex[i]].
                accepted=0;
          }

        }
      }
    }
  }
}


void CDF() {
  int* mark = (int*)malloc(sizeof(int)*
      processesCount);
  int dayCounting = 0;
  Process* rawDay = (Process*)malloc(
      sizeof(Process)*processesCount);
  int* acceptedIndex = (int*)malloc(
      sizeof(int)*processesCount);
  int i;
  memset(mark, 0, sizeof(int)*
      processesCount);
  for (i = 0; i < processesCount; i++) {
    int min = 0x7fffffff;
    int minIndex = -1;
    for (int j = 0; j < processesCount;
        j++) {
      if (mark[j] == 0 && processes[j].
          dueDate < min) {
        min = processes[j].dueDate;
        minIndex = j;
      }
    }
    mark[minIndex] = 1;
    acceptedIndex[dayCounting] =
        minIndex;
    rawDay[dayCounting++] = processes[
        minIndex];
  }

  int j,k;
  int avaliableDays=endPeiod;
  int XDays=endPeiod;
  int YDays=endPeiod;
  int ZDays=endPeiod;
int currentDay=0;

int XYZStatus[3]={0,0,0};
for(i=0;i<processesCount;i++){
  int productivity=0;
  for(k=0;k<3;k++){
    while(XYZStatus[0]!=0&&XYZStatus
        [1]!=0&&XYZStatus[2]!=0){
      XYZStatus[0]--;
      XYZStatus[1]--;
      XYZStatus[2]--;
      currentDay++;
    }
    if (XYZStatus[k]==0){
      if(k==0){

        productivity=300;
        int needDays=rawDay[i].
            quantity/productivity;
        int jugde=rawDay[i].quantity%
            productivity;
        if(jugde!=0){
          needDays++;
        }
        if(needDays>rawDay[i].dueDate-
            currentDay){
          processes[acceptedIndex[i]].
              accepted=0;
        }
        else if(rawDay[i].dueDate-
            currentDay>=needDays&&
            needDays<=XDays){
          XYZStatus[k]=needDays;
          XDays-=needDays;
          processes[acceptedIndex[i]].
              accepted=1;
          for(j=0;j<needDays;j++){

            day[k][dayCount[k]].
                Product=rawDay[i];
            day[k][dayCount[k]].
                producedQuantity=
                productivity;
            if(jugde && j==needDays-1)
                {
              day[k][dayCount[k]].
                  producedQuantity=
                  jugde;
            }
            dayCount[k]++;
          }

          break;

        }else{
          processes[acceptedIndex[i]].
              accepted=0;
        }

      }else if(k==1){

        productivity=400;
        int needDays=rawDay[i].
            quantity/productivity;
        int jugde=rawDay[i].quantity%
            productivity;
        if(jugde!=0){
          needDays++;
        }
        if(needDays>rawDay[i].dueDate-
            currentDay){
          processes[acceptedIndex[i]].
              accepted=0;
        }
        if(rawDay[i].dueDate-
            currentDay>=needDays&&
            needDays<=YDays){
          XYZStatus[k]=needDays;
          YDays-=needDays;
          processes[acceptedIndex[i]].
              accepted=1;
          for(j=0;j<needDays;j++){
            day[k][dayCount[k]].
                Product=rawDay[i];
            day[k][dayCount[k]].
                producedQuantity=
                productivity;
            if(jugde && j==needDays-1)
                {
              day[k][dayCount[k]].
                  producedQuantity=
                  jugde;
            }
            dayCount[k]++;
          }
          break;

        }else{
          processes[acceptedIndex[i]].
              accepted=0;
        }

      }else if(k==2){

        productivity=500;
        int needDays=rawDay[i].
            quantity/productivity;
        int jugde=rawDay[i].quantity%
            productivity;
        if(jugde!=0){
          needDays++;
        }
```

```
          if(needDays>rawDay[i].dueDate-
              currentDay){
            processes[acceptedIndex[i]].
                accepted=0;
          }
          else if(rawDay[i].dueDate-
              currentDay>=needDays&&
              needDays<=ZDays){
            XYZStatus[k]=needDays;
            ZDays-=needDays;
            processes[acceptedIndex[i]].
                accepted=1;
            for(j=0;j<needDays;j++){
              day[k][dayCount[k]].
                  Product=rawDay[i];
              day[k][dayCount[k]].
                  producedQuantity=
                  productivity;
              if(jugde && j==needDays-1)
                  {
                day[k][dayCount[k]].
                    producedQuantity=
                    jugde;
              }
              dayCount[k]++;
            }
            break;

          }else{
            processes[acceptedIndex[i]].
                accepted=0;
          }

        }
      }
    }
  }
}

void runPLS(int alg) {
  switch (alg) {


  case 0:
    FCFS();
    break;
  case 1:
    priorityScheduling();
    break;
  case 2:
    CDF();
    break;
  }
}
```

G. RUNPLS.H

```
#include "tools.h"

void runPLS(int alg);

// void algPR(...)
// void algFCFS(...)
// void algSJF(...)
```

H. TOOLS.C

```
#include "tools.h"

//#define _DEBUG_
#ifdef _DEBUG_
Process processes[10000];
DayArrange day[10000];
int endPeiod;
time_t startPeiod;

int main(){
}
#endif

int streq(const char* a, const char* b)
    {
  if (strlen(a) != strlen(b)) return 0;
  return !memcmp(a, b, sizeof(char) *
      strlen(b));
}

char** genCommand(char* str, int* len) {
  char** result = malloc(10*sizeof(char
      *));
  int l = 0;

  if (str[strlen(str) - 1] == '\n' ||
      str[strlen(str) - 1] == '\r')
    str[strlen(str) - 1] = 0; // remove
        the \r
  char *token = strtok(str, " ");
  while( token != NULL ) {
    result[l++] = token;
    token = strtok(NULL, " ");
  }
  *len = l;
  return result;
}

int checkCommand(char* str) {
  if (streq(str, "addPEIOD"))
    return 0;
  if (streq(str, "addORDER"))
    return 1;
  if (streq(str, "addBATCH"))
    return 2;
  if (streq(str, "runPLS"))
    return 3;
  if (streq(str, "exitPLS"))
    return 4;
  if (streq(str, "addPERIOD"))
    return 0;
  return -1;
}

int commandAlg(char* alg) {
  if (streq(alg, "FCFS"))
    return 0;
  if (streq(alg, "PR"))
    return 1;
  if (streq(alg, "CDF"))
    return 2;
  return -1;
}

void initTime(char* startTime) {
  struct tm tm;
  memset(&tm, 0, sizeof(tm));
  strptime(startTime, "%Y-%m-%d", &tm);
  startPeiod = mktime(&tm);
}

int timeToInt(char* str) {
  struct tm tm;
  memset(&tm, 0, sizeof(tm));
  strptime(str, "%Y-%m-%d", &tm);
  return (int)(difftime(mktime(&tm),
      startPeiod)/86400 + 0.5);
}

char* intToTime(int i) {
  char* str = malloc(20*sizeof(char));
  struct tm tm = *(localtime(&startPeiod
      ));
  tm.tm_mday += i;
  mktime(&tm);
  strftime(str, 20, "%Y-%m-%d", &tm);
  return str;
}

void printMenu(){
  printf("Please enter:\n> ");
}

void errorAlg(char* str) {
  printf("runPLS: %s: algorithm not
      found\n", str);
}

void errorCommand(char* str) {
  printf("%s: command not found\n", str)
      ;
}
```

```
void errorUsage(int c) {
  switch (c) {
  case 0:
    printf("Usage: addPERIOD start_date
        end_date\n");
    printf("specify the period for
        scheduling the production\n");
  case 1:
    printf("Usage: addORDER order_number
        due_date quantity product_name\n
        ");
    printf("add an order and the details
        to the scheduler.\n");
  case 2:
    printf("Usage: addBATCH filename\n")
        ;
    printf("input multiple orders in one
        batch file.\n");
  case 3:
    printf("Usage: runPLS algorithm |
        printREPORT [> filename]\n");
    printf("generate a schedule with the
        specified algorithm.\n");
  }
}

int checkRunUsage (char** c, int l) {
  if (l < 4) return l == 2;
  if (!streq(c[2], "|") || !streq(c[3],
      "printREPORT")) return 0;
  if (l < 6) return l == 4;
  if (!streq(c[4], ">")) return 0;
  return l == 6;
}
```

I. TOOLS.H

```
#ifndef TOOLS_H
#define TOOLS_H

#define _XOPEN_SOURCE

#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>

typedef struct proc {
  char orderNumber[100];
  int dueDate;
  int quantity;
  int products;
  int categorie;
  int accepted; // modify by runpls
  int plantX; // the days of each plant
      used
  int plantY;
  int plantZ;
} Process;


typedef struct dayArrange {
  Process Product;
  int producedQuantity;
} DayArrange;

extern Process processes[10000];
extern int processesCount;
extern DayArrange day[3][10000];
extern int dayCount[3];
extern time_t startPeiod;
extern int endPeiod;


int streq(const char* a, const char* b);

void printMenu();
void errorCommand(char* str);
void errorAlg(char* str);
void errorUsage(int c);

char** genCommand(char* str , int* len);

int checkCommand(char* str);
int commandAlg(char* str);
int checkRunUsage(char** command, int
    len);

// the int is the day since startpeiod
// startpeiod is 0, and so on
// you can call initTime to set
    startpeiod
void initTime(char* startTime);
int timeToInt(char* str);
char* intToTime(int i);

#endif
```

APPENDIX B SAMPLE OUTPUTS

A. REPORT_CDF.TXT

```
***PLS Schedule Analysis Report***

Algorithm used: CDF

There are 24 Orders ACCEPTED. Details
    are as follows:
ORDER NUMBER      START              END
           DAYS      QUANTITY            PLANT
=========================================
    ======================================
    ====
P0055              2024-06-01
    2024-06-04          4          1200
             PLANT_X
P0019              2024-06-05
    2024-06-06          2          400
             PLANT_X
P0090              2024-06-07
    2024-06-11          5          1500
             PLANT_X
P0037              2024-06-12
    2024-06-16          5          1300
             PLANT_X
P0006              2024-06-17
    2024-06-18          2          400
             PLANT_X
P0092              2024-06-19
    2024-06-28          10         3000
             PLANT_X
P0085              2024-06-29
    2024-06-29          1          300
             PLANT_X
P0094              2024-06-01
    2024-06-03          3          1000
             PLANT_Y
P0022              2024-06-04
    2024-06-06          3          900
             PLANT_Y
P0036              2024-06-07
    2024-06-12          6          2400
             PLANT_Y
P0008              2024-06-13
    2024-06-19          7          2800
             PLANT_Y
P0068              2024-06-20
    2024-06-24          5          1900
             PLANT_Y
P0083              2024-06-25
    2024-06-27          3          1100
             PLANT_Y
P0053              2024-06-28
    2024-06-28          1          200
             PLANT_Y
P0063              2024-06-29
    2024-06-29          1          400
             PLANT_Y
P0100              2024-06-01
    2024-06-01          1          500
             PLANT_Z
P0069              2024-06-02
    2024-06-04          3          1100
             PLANT_Z
P0015              2024-06-05
    2024-06-06          2          700
             PLANT_Z
P0095              2024-06-07
    2024-06-14          8          3600
```

```
            PLANT_Z
P0024                2024–06–15
    2024–06–16        2          600
            PLANT_Z
P0017                2024–06–17
    2024–06–17        1          500
            PLANT_Z
P0091                2024–06–18
    2024–06–27        10         4700
            PLANT_Z
P0081                2024–06–28
    2024–06–28        1          300
            PLANT_Z
P0002                2024–06–29
    2024–06–29        1          500
            PLANT_Z
        – END –

========================================
    ====================================
    ====
There are 76 Orders REJECTED. Details
    are as follows:
ORDER NUMBER     PRODUCT NAME     Due Date
            QUANTITY
========================================
    ====================================
    ====
P0001                Product_E
    2024–06–25        4800
P0003                Product_A
    2024–07–10        3900
P0004                Product_B
    2024–08–09        3200
P0005                Product_C
    2024–07–29        700
P0007                Product_H
    2024–09–02        400
P0009                Product_I
    2024–07–17        2600
P0010                Product_H
    2024–06–09        3400
P0011                Product_G
    2024–07–20        2600
P0012                Product_D
    2024–06–08        3400
P0013                Product_E
    2024–08–21        800
P0014                Product_A
    2024–08–09        5000
P0016                Product_D
    2024–08–13        2600
P0018                Product_A
    2024–09–20        2000
P0020                Product_B
    2024–08–26        5000
```

```
P0021                Product_C
    2024–09–09        3500
P0023                Product_A
    2024–07–30        4700
P0025                Product_B
    2024–08–02        4600
P0026                Product_E
    2024–07–24        900
P0027                Product_C
    2024–09–09        1600
P0028                Product_A
    2024–08–12        2200
P0029                Product_F
    2024–08–23        2000
P0030                Product_F
    2024–07–12        700
P0031                Product_C
    2024–07–29        2300
P0032                Product_F
    2024–09–27        100
P0033                Product_A
    2024–08–23        3100
P0034                Product_H
    2024–06–21        4600
P0035                Product_C
    2024–07–26        700
P0038                Product_B
    2024–08–10        3900
P0039                Product_C
    2024–07–11        3000
P0040                Product_D
    2024–09–02        4600
P0041                Product_E
    2024–08–16        1300
P0042                Product_D
    2024–08–22        200
P0043                Product_G
    2024–08–28        4700
P0044                Product_D
    2024–07–30        2300
P0045                Product_C
    2024–09–01        1600
P0046                Product_A
    2024–09–02        2000
P0047                Product_B
    2024–07–21        4500
P0048                Product_D
    2024–06–22        3100
P0049                Product_I
    2024–07–21        2600
P0050                Product_F
    2024–08–09        800
P0051                Product_H
    2024–06–07        1700
P0052                Product_I
    2024–06–17        2200
```

```
P0054                Product_F
    2024-08-30           2600
P0056                Product_C
    2024-08-26           100
P0057                Product_F
    2024-08-05           4800
P0058                Product_A
    2024-06-21           4800
P0059                Product_I
    2024-06-11           2500
P0060                Product_E
    2024-08-27           3500
P0061                Product_C
    2024-06-18           3800
P0062                Product_D
    2024-07-25           3400
P0064                Product_D
    2024-06-06           3900
P0065                Product_B
    2024-07-31           3300
P0066                Product_D
    2024-08-17           1700
P0067                Product_F
    2024-09-12           2200
P0070                Product_B
    2024-09-04           800
P0071                Product_A
    2024-07-17           2600
P0072                Product_C
    2024-07-12           4000
P0073                Product_G
    2024-07-26           2200
P0074                Product_I
    2024-07-10           2700
P0075                Product_E
    2024-08-18           600
P0076                Product_B
    2024-07-29           3300
P0077                Product_I
    2024-07-17           2600
P0078                Product_B
    2024-08-31           4400
P0079                Product_D
    2024-08-21           4700
P0080                Product_G
    2024-08-10           3400
P0082                Product_I
    2024-07-05           5000
P0084                Product_G
    2024-07-12           1600
P0086                Product_A
    2024-08-11           3000
P0087                Product_G
    2024-08-20           3000
P0088                Product_I
    2024-08-27           1300
```

```
P0089                Product_I
    2024-06-13           3300
P0093                Product_A
    2024-06-06           2500
P0096                Product_I
    2024-07-23           1100
P0097                Product_C
    2024-07-18           1900
P0098                Product_D
    2024-07-31           1200
P0099                Product_A
    2024-06-18           4200
          - END -

========================================
    ======================================
    ====
Plant_X
Date                 Product  Name       Order
    Number       Quantity(Produced)
    DueDate
2024-06-01           Product_H           P0055
             300
    2024-06-05
2024-06-02           Product_H           P0055
             300
    2024-06-05
2024-06-03           Product_H           P0055
             300
    2024-06-05
2024-06-04           Product_H           P0055
             300
    2024-06-05
2024-06-05           Product_D           P0019
             300
    2024-06-09
2024-06-06           Product_D           P0019
             100
    2024-06-09
2024-06-07           Product_B           P0090
             300
    2024-06-16
2024-06-08           Product_B           P0090
             300
    2024-06-16
2024-06-09           Product_B           P0090
             300
    2024-06-16
2024-06-10           Product_B           P0090
             300
    2024-06-16
2024-06-11           Product_B           P0090
             300
    2024-06-16
2024-06-12           Product_C           P0037
             300
    2024–06–19
2024–06–13        Product_C        P0037
         300
    2024–06–19
2024–06–14        Product_C        P0037
         300
    2024–06–19
2024–06–15        Product_C        P0037
         300
    2024–06–19
2024–06–16        Product_C        P0037
         100
    2024–06–19
2024–06–17        Product_A        P0006
         300
    2024–06–28
2024–06–18        Product_A        P0006
         100
    2024–06–28
2024–06–19        Product_F        P0092
         300
    2024–06–30
2024–06–20        Product_F        P0092
         300
    2024–06–30
2024–06–21        Product_F        P0092
         300
    2024–06–30
2024–06–22        Product_F        P0092
         300
    2024–06–30
2024–06–23        Product_F        P0092
         300
    2024–06–30
2024–06–24        Product_F        P0092
         300
    2024–06–30
2024–06–25        Product_F        P0092
         300
    2024–06–30
2024–06–26        Product_F        P0092
         300
    2024–06–30
2024–06–27        Product_F        P0092
         300
    2024–06–30
2024–06–28        Product_F        P0092
         300
    2024–06–30
2024–06–29        Product_I        P0085
         300
    2024–07–27
Plant_Y
Date              Product Name     Order
    Number     Quantity(Produced)
    DueDate
```

```
2024–06–01        Product_C        P0094
         400
    2024–06–05
2024–06–02        Product_C        P0094
         400
    2024–06–05
2024–06–03        Product_C        P0094
         200
    2024–06–05
2024–06–04        Product_H        P0022
         400
    2024–06–08
2024–06–05        Product_H        P0022
         400
    2024–06–08
2024–06–06        Product_H        P0022
         100
    2024–06–08
2024–06–07        Product_F        P0036
         400
    2024–06–14
2024–06–08        Product_F        P0036
         400
    2024–06–14
2024–06–09        Product_F        P0036
         400
    2024–06–14
2024–06–10        Product_F        P0036
         400
    2024–06–14
2024–06–11        Product_F        P0036
         400
    2024–06–14
2024–06–12        Product_F        P0036
         400
    2024–06–14
2024–06–13        Product_D        P0008
         400
    2024–06–20
2024–06–14        Product_D        P0008
         400
    2024–06–20
2024–06–15        Product_D        P0008
         400
    2024–06–20
2024–06–16        Product_D        P0008
         400
    2024–06–20
2024–06–17        Product_D        P0008
         400
    2024–06–20
2024–06–18        Product_D        P0008
         400
    2024–06–20
2024–06–19        Product_D        P0008
         400
```

				2024-06-20
2024-06-20	Product_I	P0068	400	2024-07-05
2024-06-21	Product_I	P0068	400	2024-07-05
2024-06-22	Product_I	P0068	400	2024-07-05
2024-06-23	Product_I	P0068	400	2024-07-05
2024-06-24	Product_I	P0068	300	2024-07-05
2024-06-25	Product_G	P0083	400	2024-07-06
2024-06-26	Product_G	P0083	400	2024-07-06
2024-06-27	Product_G	P0083	300	2024-07-06
2024-06-28	Product_A	P0053	200	2024-07-07
2024-06-29	Product_I	P0063	400	2024-08-07

Plant_Z

Date	Product Name	Order Number	Quantity (Produced)	DueDate
2024-06-01	Product_E	P0100	500	2024-06-02
2024-06-02	Product_G	P0069	500	2024-06-06
2024-06-03	Product_G	P0069	500	2024-06-06
2024-06-04	Product_G	P0069	100	2024-06-06
2024-06-05	Product_B	P0015	500	2024-06-10
2024-06-06	Product_B	P0015	200	2024-06-10
2024-06-07	Product_C	P0095	500	2024-06-16
2024-06-08	Product_C	P0095	500	2024-06-16
2024-06-09	Product_C	P0095	500	2024-06-16
2024-06-10	Product_C	P0095	500	2024-06-16
2024-06-11	Product_C	P0095	500	2024-06-16
2024-06-12	Product_C	P0095	500	2024-06-16
2024-06-13	Product_C	P0095	500	2024-06-16
2024-06-14	Product_C	P0095	100	2024-06-16
2024-06-15	Product_B	P0024	500	2024-06-22
2024-06-16	Product_B	P0024	100	2024-06-22
2024-06-17	Product_B	P0017	500	2024-06-28
2024-06-18	Product_A	P0091	500	2024-06-30
2024-06-19	Product_A	P0091	500	2024-06-30
2024-06-20	Product_A	P0091	500	2024-06-30
2024-06-21	Product_A	P0091	500	2024-06-30
2024-06-22	Product_A	P0091	500	2024-06-30
2024-06-23	Product_A	P0091	500	2024-06-30
2024-06-24	Product_A	P0091	500	2024-06-30
2024-06-25	Product_A	P0091	500	2024-06-30
2024-06-26	Product_A	P0091	500	

```
    2024-06-30
2024-06-27          Product_A           P0091
          200
    2024-06-30
2024-06-28          Product_F           P0081
          300
    2024-07-09
2024-06-29          Product_H           P0002
          500
    2024-08-17

                   - End -

========================================
    ====================================
    ====

***PERFORMANCE

Plant X:
        Number of days in use:
                29 days
        Number of products produced:
                8100 (in total)
        Utilization of the plant:
                93.0 %

Plant Y:
        Number of days in use:
                29 days
        Number of products produced:
                10700 (in total)
        Utilization of the plant:
                92.0 %

Plant Z:
        Number of days in use:
                29 days
        Number of products produced:
                12500 (in total)
        Utilization of the plant:
                86.0 %

Overall of utilization:
            89.0 %
```

B. REPORT_FCFS.TXT

```
***PLS Schedule Analysis Report***

Algorithm used: FCFS

There are 20 Orders ACCEPTED. Details
    are as follows:
ORDER NUMBER      START            END
          DAYS     QUANTITY          PLANT
========================================
    ====================================
    ====
P0001               2024-06-01
    2024-06-16          16         4800
            PLANT_X
P0011               2024-06-17
    2024-06-25          9          2600
            PLANT_X
P0030               2024-06-26
    2024-06-28          3          700
            PLANT_X
P0042               2024-06-29
    2024-06-29          1          200
            PLANT_X
P0002               2024-06-01
    2024-06-02          2          500
            PLANT_Y
P0004               2024-06-03
    2024-06-10          8          3200
            PLANT_Y
P0006               2024-06-11
    2024-06-11          1          400
            PLANT_Y
P0008               2024-06-12
    2024-06-18          7          2800
            PLANT_Y
P0016               2024-06-19
    2024-06-25          7          2600
            PLANT_Y
P0027               2024-06-26
    2024-06-29          4          1600
            PLANT_Y
P0003               2024-06-01
    2024-06-08          8          3900
            PLANT_Z
P0005               2024-06-09
    2024-06-10          2          700
            PLANT_Z
P0007               2024-06-11
    2024-06-11          1          400
            PLANT_Z
P0009               2024-06-12
    2024-06-17          6          2600
            PLANT_Z
P0013               2024-06-18
    2024-06-19          2          800
            PLANT_Z
P0017               2024-06-20
    2024-06-20          1          500
            PLANT_Z
P0018               2024-06-21
    2024-06-24          4          2000
            PLANT_Z
P0026               2024-06-25
    2024-06-26          2          900
            PLANT_Z
```

```
P0032              2024–06–27
    2024–06–27          1          100
             PLANT_Z
P0035              2024–06–28
    2024–06–29          2          700
             PLANT_Z
         – END –

==========================================
    ======================================
    ====
There  are  80  Orders  REJECTED.  Details
    are  as  follows:
ORDER NUMBER      PRODUCT NAME       Due  Date
             QUANTITY
==========================================
    ======================================
    ====
P0010              Product_H
    2024–06–09         3400
P0012              Product_D
    2024–06–08         3400
P0014              Product_A
    2024–08–09         5000
P0015              Product_B
    2024–06–10         700
P0019              Product_D
    2024–06–09         400
P0020              Product_B
    2024–08–26         5000
P0021              Product_C
    2024–09–09         3500
P0022              Product_H
    2024–06–08         900
P0023              Product_A
    2024–07–30         4700
P0024              Product_B
    2024–06–22         600
P0025              Product_B
    2024–08–02         4600
P0028              Product_A
    2024–08–12         2200
P0029              Product_F
    2024–08–23         2000
P0031              Product_C
    2024–07–29         2300
P0033              Product_A
    2024–08–23         3100
P0034              Product_H
    2024–06–21         4600
P0036              Product_F
    2024–06–14         2400
P0037              Product_C
    2024–06–19         1300
P0038              Product_B
    2024–08–10         3900
P0039              Product_C
    2024–07–11         3000
P0040              Product_D
    2024–09–02         4600
P0041              Product_E
    2024–08–16         1300
P0043              Product_G
    2024–08–28         4700
P0044              Product_D
    2024–07–30         2300
P0045              Product_C
    2024–09–01         1600
P0046              Product_A
    2024–09–02         2000
P0047              Product_B
    2024–07–21         4500
P0048              Product_D
    2024–06–22         3100
P0049              Product_I
    2024–07–21         2600
P0050              Product_F
    2024–08–09         800
P0051              Product_H
    2024–06–07         1700
P0052              Product_I
    2024–06–17         2200
P0053              Product_A
    2024–07–07         200
P0054              Product_F
    2024–08–30         2600
P0055              Product_H
    2024–06–05         1200
P0056              Product_C
    2024–08–26         100
P0057              Product_F
    2024–08–05         4800
P0058              Product_A
    2024–06–21         4800
P0059              Product_I
    2024–06–11         2500
P0060              Product_E
    2024–08–27         3500
P0061              Product_C
    2024–06–18         3800
P0062              Product_D
    2024–07–25         3400
P0063              Product_I
    2024–08–07         400
P0064              Product_D
    2024–06–06         3900
P0065              Product_B
    2024–07–31         3300
P0066              Product_D
    2024–08–17         1700
P0067              Product_F
    2024–09–12         2200
```

```
P0068              Product_I
    2024-07-05       1900
P0069              Product_G
    2024-06-06       1100
P0070              Product_B
    2024-09-04       800
P0071              Product_A
    2024-07-17       2600
P0072              Product_C
    2024-07-12       4000
P0073              Product_G
    2024-07-26       2200
P0074              Product_I
    2024-07-10       2700
P0075              Product_E
    2024-08-18       600
P0076              Product_B
    2024-07-29       3300
P0077              Product_I
    2024-07-17       2600
P0078              Product_B
    2024-08-31       4400
P0079              Product_D
    2024-08-21       4700
P0080              Product_G
    2024-08-10       3400
P0081              Product_F
    2024-07-09       300
P0082              Product_I
    2024-07-05       5000
P0083              Product_G
    2024-07-06       1100
P0084              Product_G
    2024-07-12       1600
P0085              Product_I
    2024-07-27       300
P0086              Product_A
    2024-08-11       3000
P0087              Product_G
    2024-08-20       3000
P0088              Product_I
    2024-08-27       1300
P0089              Product_I
    2024-06-13       3300
P0090              Product_B
    2024-06-16       1500
P0091              Product_A
    2024-06-30       4700
P0092              Product_F
    2024-06-30       3000
P0093              Product_A
    2024-06-06       2500
P0094              Product_C
    2024-06-05       1000
P0095              Product_C
    2024-06-16       3600
```

```
P0096              Product_I
    2024-07-23       1100
P0097              Product_C
    2024-07-18       1900
P0098              Product_D
    2024-07-31       1200
P0099              Product_A
    2024-06-18       4200
P0100              Product_E
    2024-06-02       500
          - END -

================================================
    ============================================
    ====
Plant_X
Date               Product Name       Order
    Number     Quantity(Produced)
    DueDate
2024-06-01         Product_E          P0001
              300
    2024-06-25
2024-06-02         Product_E          P0001
              300
    2024-06-25
2024-06-03         Product_E          P0001
              300
    2024-06-25
2024-06-04         Product_E          P0001
              300
    2024-06-25
2024-06-05         Product_E          P0001
              300
    2024-06-25
2024-06-06         Product_E          P0001
              300
    2024-06-25
2024-06-07         Product_E          P0001
              300
    2024-06-25
2024-06-08         Product_E          P0001
              300
    2024-06-25
2024-06-09         Product_E          P0001
              300
    2024-06-25
2024-06-10         Product_E          P0001
              300
    2024-06-25
2024-06-11         Product_E          P0001
              300
    2024-06-25
2024-06-12         Product_E          P0001
              300
    2024-06-25
2024-06-13         Product_E          P0001
```

```
          300
     2024–06–25
2024–06–14        Product_E        P0001
          300
     2024–06–25
2024–06–15        Product_E        P0001
          300
     2024–06–25
2024–06–16        Product_E        P0001
          300
     2024–06–25
2024–06–17        Product_G        P0011
          300
     2024–07–20
2024–06–18        Product_G        P0011
          300
     2024–07–20
2024–06–19        Product_G        P0011
          300
     2024–07–20
2024–06–20        Product_G        P0011
          300
     2024–07–20
2024–06–21        Product_G        P0011
          300
     2024–07–20
2024–06–22        Product_G        P0011
          300
     2024–07–20
2024–06–23        Product_G        P0011
          300
     2024–07–20
2024–06–24        Product_G        P0011
          300
     2024–07–20
2024–06–25        Product_G        P0011
          200
     2024–07–20
2024–06–26        Product_F        P0030
          300
     2024–07–12
2024–06–27        Product_F        P0030
          300
     2024–07–12
2024–06–28        Product_F        P0030
          100
     2024–07–12
2024–06–29        Product_D        P0042
          200
     2024–08–22
Plant_Y
Date              Product  Name    Order
     Number    Quantity(Produced)
     DueDate
2024–06–01        Product_H        P0002
          400
```

```
     2024–08–17
2024–06–02        Product_H        P0002
          100
     2024–08–17
2024–06–03        Product_B        P0004
          400
     2024–08–09
2024–06–04        Product_B        P0004
          400
     2024–08–09
2024–06–05        Product_B        P0004
          400
     2024–08–09
2024–06–06        Product_B        P0004
          400
     2024–08–09
2024–06–07        Product_B        P0004
          400
     2024–08–09
2024–06–08        Product_B        P0004
          400
     2024–08–09
2024–06–09        Product_B        P0004
          400
     2024–08–09
2024–06–10        Product_B        P0004
          400
     2024–08–09
2024–06–11        Product_A        P0006
          400
     2024–06–28
2024–06–12        Product_D        P0008
          400
     2024–06–20
2024–06–13        Product_D        P0008
          400
     2024–06–20
2024–06–14        Product_D        P0008
          400
     2024–06–20
2024–06–15        Product_D        P0008
          400
     2024–06–20
2024–06–16        Product_D        P0008
          400
     2024–06–20
2024–06–17        Product_D        P0008
          400
     2024–06–20
2024–06–18        Product_D        P0008
          400
     2024–06–20
2024–06–19        Product_D        P0016
          400
     2024–08–13
2024–06–20        Product_D        P0016
```

```
           400
     2024–08–13
2024–06–21       Product_D       P0016
           400
     2024–08–13
2024–06–22       Product_D       P0016
           400
     2024–08–13
2024–06–23       Product_D       P0016
           400
     2024–08–13
2024–06–24       Product_D       P0016
           400
     2024–08–13
2024–06–25       Product_D       P0016
           200
     2024–08–13
2024–06–26       Product_C       P0027
           400
     2024–09–09
2024–06–27       Product_C       P0027
           400
     2024–09–09
2024–06–28       Product_C       P0027
           400
     2024–09–09
2024–06–29       Product_C       P0027
           400
     2024–09–09
Plant_Z
Date             Product  Name   Order
     Number    Quantity(Produced)
     DueDate
2024–06–01       Product_A       P0003
           500
     2024–07–10
2024–06–02       Product_A       P0003
           500
     2024–07–10
2024–06–03       Product_A       P0003
           500
     2024–07–10
2024–06–04       Product_A       P0003
           500
     2024–07–10
2024–06–05       Product_A       P0003
           500
     2024–07–10
2024–06–06       Product_A       P0003
           500
     2024–07–10
2024–06–07       Product_A       P0003
           500
     2024–07–10
2024–06–08       Product_A       P0003
           400
```

```
     2024–07–10
2024–06–09       Product_C       P0005
           500
     2024–07–29
2024–06–10       Product_C       P0005
           200
     2024–07–29
2024–06–11       Product_H       P0007
           400
     2024–09–02
2024–06–12       Product_I       P0009
           500
     2024–07–17
2024–06–13       Product_I       P0009
           500
     2024–07–17
2024–06–14       Product_I       P0009
           500
     2024–07–17
2024–06–15       Product_I       P0009
           500
     2024–07–17
2024–06–16       Product_I       P0009
           500
     2024–07–17
2024–06–17       Product_I       P0009
           100
     2024–07–17
2024–06–18       Product_E       P0013
           500
     2024–08–21
2024–06–19       Product_E       P0013
           300
     2024–08–21
2024–06–20       Product_B       P0017
           500
     2024–06–28
2024–06–21       Product_A       P0018
           500
     2024–09–20
2024–06–22       Product_A       P0018
           500
     2024–09–20
2024–06–23       Product_A       P0018
           500
     2024–09–20
2024–06–24       Product_A       P0018
           500
     2024–09–20
2024–06–25       Product_E       P0026
           500
     2024–07–24
2024–06–26       Product_E       P0026
           400
     2024–07–24
2024–06–27       Product_F       P0032
```

```
           100
   2024-09-27
2024-06-28        Product_C        P0035
           500
   2024-07-26
2024-06-29        Product_C        P0035
           200
   2024-07-26

                    - End -

========================================
   ========================================
   ====

***PERFORMANCE

Plant X:
        Number of days in use:
                    29 days
        Number of products produced:
                    8300 (in total)
        Utilization of the plant:
                    95.0 %

Plant Y:
        Number of days in use:
                    29 days
        Number of products produced:
                    11100 (in total)
        Utilization of the plant:
                    95.0 %

Plant Z:
        Number of days in use:
                    29 days
        Number of products produced:
                    12600 (in total)
        Utilization of the plant:
                    86.0 %

Overall of utilization:
             91.0 %
```

C. REPORT_PR.TXT

```
***PLS Schedule Analysis Report***

Algorithm used: FCFS

There are 20 Orders ACCEPTED. Details
   are as follows:
ORDER NUMBER    START            END
           DAYS     QUANTITY          PLANT
========================================
   ========================================
   ====
P0001            2024-06-01
   2024-06-16     16       4800
          PLANT_X
P0011            2024-06-17
   2024-06-25     9        2600
          PLANT_X
P0030            2024-06-26
   2024-06-28     3        700
          PLANT_X
P0042            2024-06-29
   2024-06-29     1        200
          PLANT_X
P0002            2024-06-01
   2024-06-02     2        500
          PLANT_Y
P0004            2024-06-03
   2024-06-10     8        3200
          PLANT_Y
P0006            2024-06-11
   2024-06-11     1        400
          PLANT_Y
P0008            2024-06-12
   2024-06-18     7        2800
          PLANT_Y
P0016            2024-06-19
   2024-06-25     7        2600
          PLANT_Y
P0027            2024-06-26
   2024-06-29     4        1600
          PLANT_Y
P0003            2024-06-01
   2024-06-08     8        3900
          PLANT_Z
P0005            2024-06-09
   2024-06-10     2        700
          PLANT_Z
P0007            2024-06-11
   2024-06-11     1        400
          PLANT_Z
P0009            2024-06-12
   2024-06-17     6        2600
          PLANT_Z
P0013            2024-06-18
   2024-06-19     2        800
          PLANT_Z
P0017            2024-06-20
   2024-06-20     1        500
          PLANT_Z
P0018            2024-06-21
   2024-06-24     4        2000
          PLANT_Z
P0026            2024-06-25
   2024-06-26     2        900
          PLANT_Z
P0032            2024-06-27
   2024-06-27     1        100
```

```
              PLANT_Z
P0035               2024-06-28
    2024-06-29          2           700
              PLANT_Z
          - END -

==========================================
    ======================================
    ====
There are 80 Orders REJECTED. Details
    are as follows:
ORDER NUMBER    PRODUCT NAME     Due Date
              QUANTITY
==========================================
    ======================================
    ====
P0010               Product_H
    2024-06-09          3400
P0012               Product_D
    2024-06-08          3400
P0014               Product_A
    2024-08-09          5000
P0015               Product_B
    2024-06-10          700
P0019               Product_D
    2024-06-09          400
P0020               Product_B
    2024-08-26          5000
P0021               Product_C
    2024-09-09          3500
P0022               Product_H
    2024-06-08          900
P0023               Product_A
    2024-07-30          4700
P0024               Product_B
    2024-06-22          600
P0025               Product_B
    2024-08-02          4600
P0028               Product_A
    2024-08-12          2200
P0029               Product_F
    2024-08-23          2000
P0031               Product_C
    2024-07-29          2300
P0033               Product_A
    2024-08-23          3100
P0034               Product_H
    2024-06-21          4600
P0036               Product_F
    2024-06-14          2400
P0037               Product_C
    2024-06-19          1300
P0038               Product_B
    2024-08-10          3900
P0039               Product_C
    2024-07-11          3000
P0040               Product_D
    2024-09-02          4600
P0041               Product_E
    2024-08-16          1300
P0043               Product_G
    2024-08-28          4700
P0044               Product_D
    2024-07-30          2300
P0045               Product_C
    2024-09-01          1600
P0046               Product_A
    2024-09-02          2000
P0047               Product_B
    2024-07-21          4500
P0048               Product_D
    2024-06-22          3100
P0049               Product_I
    2024-07-21          2600
P0050               Product_F
    2024-08-09          800
P0051               Product_H
    2024-06-07          1700
P0052               Product_I
    2024-06-17          2200
P0053               Product_A
    2024-07-07          200
P0054               Product_F
    2024-08-30          2600
P0055               Product_H
    2024-06-05          1200
P0056               Product_C
    2024-08-26          100
P0057               Product_F
    2024-08-05          4800
P0058               Product_A
    2024-06-21          4800
P0059               Product_I
    2024-06-11          2500
P0060               Product_E
    2024-08-27          3500
P0061               Product_C
    2024-06-18          3800
P0062               Product_D
    2024-07-25          3400
P0063               Product_I
    2024-08-07          400
P0064               Product_D
    2024-06-06          3900
P0065               Product_B
    2024-07-31          3300
P0066               Product_D
    2024-08-17          1700
P0067               Product_F
    2024-09-12          2200
P0068               Product_I
    2024-07-05          1900
```

```
P0069               Product_G
    2024-06-06        1100
P0070               Product_B
    2024-09-04        800
P0071               Product_A
    2024-07-17        2600
P0072               Product_C
    2024-07-12        4000
P0073               Product_G
    2024-07-26        2200
P0074               Product_I
    2024-07-10        2700
P0075               Product_E
    2024-08-18        600
P0076               Product_B
    2024-07-29        3300
P0077               Product_I
    2024-07-17        2600
P0078               Product_B
    2024-08-31        4400
P0079               Product_D
    2024-08-21        4700
P0080               Product_G
    2024-08-10        3400
P0081               Product_F
    2024-07-09        300
P0082               Product_I
    2024-07-05        5000
P0083               Product_G
    2024-07-06        1100
P0084               Product_G
    2024-07-12        1600
P0085               Product_I
    2024-07-27        300
P0086               Product_A
    2024-08-11        3000
P0087               Product_G
    2024-08-20        3000
P0088               Product_I
    2024-08-27        1300
P0089               Product_I
    2024-06-13        3300
P0090               Product_B
    2024-06-16        1500
P0091               Product_A
    2024-06-30        4700
P0092               Product_F
    2024-06-30        3000
P0093               Product_A
    2024-06-06        2500
P0094               Product_C
    2024-06-05        1000
P0095               Product_C
    2024-06-16        3600
P0096               Product_I
    2024-07-23        1100
```

```
P0097               Product_C
    2024-07-18        1900
P0098               Product_D
    2024-07-31        1200
P0099               Product_A
    2024-06-18        4200
P0100               Product_E
    2024-06-02        500
          - END -

================================================
    ============================================
    ====
Plant_X
Date                Product Name        Order
    Number      Quantity(Produced)
    DueDate
2024-06-01          Product_E           P0001
                300
    2024-06-25
2024-06-02          Product_E           P0001
                300
    2024-06-25
2024-06-03          Product_E           P0001
                300
    2024-06-25
2024-06-04          Product_E           P0001
                300
    2024-06-25
2024-06-05          Product_E           P0001
                300
    2024-06-25
2024-06-06          Product_E           P0001
                300
    2024-06-25
2024-06-07          Product_E           P0001
                300
    2024-06-25
2024-06-08          Product_E           P0001
                300
    2024-06-25
2024-06-09          Product_E           P0001
                300
    2024-06-25
2024-06-10          Product_E           P0001
                300
    2024-06-25
2024-06-11          Product_E           P0001
                300
    2024-06-25
2024-06-12          Product_E           P0001
                300
    2024-06-25
2024-06-13          Product_E           P0001
                300
    2024-06-25
```

```
2024-06-14        Product_E        P0001
                300
    2024-06-25
2024-06-15        Product_E        P0001
                300
    2024-06-25
2024-06-16        Product_E        P0001
                300
    2024-06-25
2024-06-17        Product_G        P0011
                300
    2024-07-20
2024-06-18        Product_G        P0011
                300
    2024-07-20
2024-06-19        Product_G        P0011
                300
    2024-07-20
2024-06-20        Product_G        P0011
                300
    2024-07-20
2024-06-21        Product_G        P0011
                300
    2024-07-20
2024-06-22        Product_G        P0011
                300
    2024-07-20
2024-06-23        Product_G        P0011
                300
    2024-07-20
2024-06-24        Product_G        P0011
                300
    2024-07-20
2024-06-25        Product_G        P0011
                200
    2024-07-20
2024-06-26        Product_F        P0030
                300
    2024-07-12
2024-06-27        Product_F        P0030
                300
    2024-07-12
2024-06-28        Product_F        P0030
                100
    2024-07-12
2024-06-29        Product_D        P0042
                200
    2024-08-22
Plant_Y
Date              Product Name     Order
    Number     Quantity(Produced)
    DueDate
2024-06-01        Product_H        P0002
                400
    2024-08-17
2024-06-02        Product_H        P0002
                100
    2024-08-17
2024-06-03        Product_B        P0004
                400
    2024-08-09
2024-06-04        Product_B        P0004
                400
    2024-08-09
2024-06-05        Product_B        P0004
                400
    2024-08-09
2024-06-06        Product_B        P0004
                400
    2024-08-09
2024-06-07        Product_B        P0004
                400
    2024-08-09
2024-06-08        Product_B        P0004
                400
    2024-08-09
2024-06-09        Product_B        P0004
                400
    2024-08-09
2024-06-10        Product_B        P0004
                400
    2024-08-09
2024-06-11        Product_A        P0006
                400
    2024-06-28
2024-06-12        Product_D        P0008
                400
    2024-06-20
2024-06-13        Product_D        P0008
                400
    2024-06-20
2024-06-14        Product_D        P0008
                400
    2024-06-20
2024-06-15        Product_D        P0008
                400
    2024-06-20
2024-06-16        Product_D        P0008
                400
    2024-06-20
2024-06-17        Product_D        P0008
                400
    2024-06-20
2024-06-18        Product_D        P0008
                400
    2024-06-20
2024-06-19        Product_D        P0016
                400
    2024-08-13
2024-06-20        Product_D        P0016
                400
    2024-08-13
```

```
2024-06-21        Product_D        P0016
                400
    2024-08-13
2024-06-22        Product_D        P0016
                400
    2024-08-13
2024-06-23        Product_D        P0016
                400
    2024-08-13
2024-06-24        Product_D        P0016
                400
    2024-08-13
2024-06-25        Product_D        P0016
                200
    2024-08-13
2024-06-26        Product_C        P0027
                400
    2024-09-09
2024-06-27        Product_C        P0027
                400
    2024-09-09
2024-06-28        Product_C        P0027
                400
    2024-09-09
2024-06-29        Product_C        P0027
                400
    2024-09-09
Plant_Z
Date              Product Name     Order
    Number      Quantity(Produced)
    DueDate
2024-06-01        Product_A        P0003
                500
    2024-07-10
2024-06-02        Product_A        P0003
                500
    2024-07-10
2024-06-03        Product_A        P0003
                500
    2024-07-10
2024-06-04        Product_A        P0003
                500
    2024-07-10
2024-06-05        Product_A        P0003
                500
    2024-07-10
2024-06-06        Product_A        P0003
                500
    2024-07-10
2024-06-07        Product_A        P0003
                500
    2024-07-10
2024-06-08        Product_A        P0003
                400
    2024-07-10
2024-06-09        Product_C        P0005
                500
    2024-07-29
2024-06-10        Product_C        P0005
                200
    2024-07-29
2024-06-11        Product_H        P0007
                400
    2024-09-02
2024-06-12        Product_I        P0009
                500
    2024-07-17
2024-06-13        Product_I        P0009
                500
    2024-07-17
2024-06-14        Product_I        P0009
                500
    2024-07-17
2024-06-15        Product_I        P0009
                500
    2024-07-17
2024-06-16        Product_I        P0009
                500
    2024-07-17
2024-06-17        Product_I        P0009
                100
    2024-07-17
2024-06-18        Product_E        P0013
                500
    2024-08-21
2024-06-19        Product_E        P0013
                300
    2024-08-21
2024-06-20        Product_B        P0017
                500
    2024-06-28
2024-06-21        Product_A        P0018
                500
    2024-09-20
2024-06-22        Product_A        P0018
                500
    2024-09-20
2024-06-23        Product_A        P0018
                500
    2024-09-20
2024-06-24        Product_A        P0018
                500
    2024-09-20
2024-06-25        Product_E        P0026
                500
    2024-07-24
2024-06-26        Product_E        P0026
                400
    2024-07-24
2024-06-27        Product_F        P0032
                100
    2024-09-27
```

```
2024−06−28          Product_C          P0035
              500
    2024−07−26
2024−06−29          Product_C          P0035
              200
    2024−07−26

                             − End −

================================================
    ============================================
    ====

***PERFORMANCE

Plant X:
         Number of days in use:
                        29 days
         Number of products produced:
                        8300 (in total)
         Utilization of the plant:
                        95.0 %

Plant Y:
         Number of days in use:
                        29 days
         Number of products produced:
                        11100 (in total)
         Utilization of the plant:
                        95.0 %

Plant Z:
         Number of days in use:
                        29 days
         Number of products produced:
                        12600 (in total)
         Utilization of the plant:
                        86.0 %

Overall of utilization:
              91.0 %
```