



# **Preparation of Papers for IEEE ACCESS**

XIAN Chong<sup>1</sup>, ZHANG Kaidi<sup>2</sup>, ZENG Junqi<sup>3</sup>, LIN Fanyue<sup>4</sup>, and YU Shijiong<sup>5</sup>

<sup>1</sup>22097486D

<sup>2</sup>22100265D

<sup>3</sup>22099611D <sup>4</sup>22099314D

<sup>5</sup>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. We his 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 re-

lationships, 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 products produced: 27700 (in total)
Utilization of the plant: 76.0 %

Plant Y:

Number of products produced: 38000 (in total)
Utilization of the plant: 78.0 %

Plant Z:

Number of products produced: 38000 (in total)
Utilization of the plant: 78.0 %

Plant Z:

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.

FIGURE 2. The result of the PR algorithm.

### **V. CONCLUSION**

This project aims to develop an application for a mediumsized 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 products produced:
Utilization of the plant:

96 days
38000 (in total)

78.0 %

Plant Z:

Number of products produced:
Utilization of the plant:

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;
                                              B. INPUT.H
  }
  processes[processesCount].dueDate =
                                              #include "tools.h"
     timeToInt(command[2]);
                                              void addPEIOD(char** command, int len);
  if (!~sscanf(command[3], "%d", &
                                              int addORDER(char** command, int len);
                                              void addBATCH(char** command, int len);
     processes [processes Count]. quantity)
     ) {
                                              C. MAIN.C
    printf("addORDER: %s: Invalid
       quantity format.\n", command[3]);
                                              #include < stdio.h>
    return -1;
                                              #include < sys/wait.h>
  }
                                              #include "tools.h"
  char str[10];
                                              #include "input.h"
                                              #include "output.h"
  int p;
  if (strlen(command[4]) != 9
                                              #include "runpls.h"
      streq(memcpy(str, command[4],
         sizeof(char) * 7), "Product_")
                                              Process processes [10000];
      | | (p = command [4][8] - 'A') >= 9
                                              int processes Count;
      || (p < 0)|
                                              DayArrange day[3][10000];
    printf("addORDER: %s: Invalid
                                              int dayCount[3];
       products.\n", command[4]);
                                              int endPeiod;
    return -1;
                                              // do not use this varible
  processes[processesCount].products = p
                                              // only used by utils of time
                                              // the usage of time function refer to
  processes[processesCount].categorie =
                                                 tools.h
                                              time_t startPeiod;
     p/3;
  processesCount ++;
                                              int main() {
  return 0;
                                                printf("\t~~WELCOME TO PLS~~\n\n");
}
                                                while (1)
                                                  printMenu();
void addBATCH(char** c, int len) {
                                                  char str [100];
  if (len != 2) return;
                                                  char* result = fgets(str, 100, stdin
  FILE* f = fopen(c[1], "r");
  if (!f) {
                                                  if (result == NULL)
    printf("addBATCH: %s: Incalid files
                                                    return 0;
                                                  int commandLen;
       .\n", c[1]);
  }
                                                  char **command = genCommand(str, &
                                                     commandLen);
  while (1)
                                                  if (commandLen == 0)
    char str [100];
                                                    continue; // why?
                                                  switch (checkCommand(command[0])) {
    char* result = fgets(str, 100, f);
    if (!result) return;
                                                  case 0:
    int commandLen;
                                                    addPEIOD(command, commandLen);
    char** command = genCommand(str, &
                                                    break;
       commandLen);
                                                  case 1:
    if (! streq(command[0], "addORDER")){
                                                    addORDER(command, commandLen);
      printf("addBATCH: Invalid command
         in files.\n");
                                                  case 2:
                                                    addBATCH(command, commandLen);
      continue;
                                                    break;
    addORDER(command, commandLen);
                                                  case 3:
```



```
if (!checkRunUsage(command,
         commandLen)) {
                                              int main()
        errorUsage(3);
        break:
                                                initTime("2023-12-30");
                                                char a[10] = "P0000";
      }
                                                for (int i = 0; i < 10; i++)
      int algTemp = commandAlg(command
          [1]);
                                                     day[i \% 3][dayCount[i \% 3]++] = (
      if (!~algTemp) {
                                                        DayArrange) {
                                                       (Process)\{"", i, 100 * i ^ 3, i\}
        errorAlg(command[1]);
                                                          \% 9, 0, 0},
        break:
                                                       100};
      runPLS(algTemp);
                                                    memcpy(day[i % 3][dayCount[i % 3]
                                                        - 1]. Product. orderNumber, a,
                                                        sizeof(a));
      if (commandLen >= 6) {
        FILE * file = fopen (command [5], "
                                                     a[4]++;
           w");
                                                  }
        printREPORT(file , algTemp);
                                                 printf("%d", day[1][0]. Product.
      } else {
        printREPORT(stdout, algTemp);
                                                    accepted);
                                                printREPORT(stdout, 0);
      memset(day, 0, sizeof(day));
      memset(dayCount, 0, sizeof(
                                              #endif
          dayCount));
                                              // Function to write 'usingdays' and '
      break:
                                                  ToTalproducedQuantity ' to the pipe
    case 4:
      printf("Bye-bye!\n");
                                              void writeToPipe(int pipe_fd, int *
                                                  usingdays, int ToTalproducedQuantity
      return 0; // exitPLS
    case -1:
                                                  []
      errorCommand(command[0]);
                                                write (pipe_fd, usingdays, size of (int)
      break;
                                                    * 3);
                                                write (pipe_fd, ToTalproducedQuantity,
    free (command);
  }
                                                    sizeof(int) * 3);
                                              }
}
D. OUTPUT.H
                                              // Function to read 'using days' and '
#include "tools.h"
                                                  ToTalproducedQuantity ' from the pipe
                                              void readFromPipe(int pipe_fd, int *
#include < stdio.h>
                                                  usingdays, int ToTalproducedQuantity
void printREPORT(FILE* file , int alg);
                                                  [])
E. OUTPUT.C
                                                read(pipe_fd, usingdays, sizeof(int) *
#include "output.h"
                                                     3);
#include <unistd.h>
                                                read (pipe_fd, ToTalproducedQuantity,
#include < sys/wait.h>
                                                    sizeof(int) * 3);
                                              }
//#define _DEBUG_ // to debug uncomment
   this line and run 'gcc output.c'
                                              // child process
#ifdef _DEBUG_
                                              // read the 'day' from pipe here
                                              // the function intToTime in 'tools.h'
Process processes [10000];
int processes Count;
                                                 may useful
DayArrange day[3][10000];
                                              void printREPORT(FILE *file , int alg)
int dayCount[3];
                                                 fprintf(file, "***PLS Schedule
int endPeiod = 30;
time t startPeiod;
                                                    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 using days [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)
        rejected Processes [
           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");
=======\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][i-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][i - 1].
                      Product.
                      orderNumber
                  intToTime(
                      startTime),
                       intToTime(
                      endTime),
                  days, quantity
                       + day[i][j
                       -11.
                      producedQuantity
                      , plant[i])
         startTime = j;
         quantity = 0;
       }
endTime = dayCount[i] - 1;
int days = endTime - startTime
    + 1;
fprintf(file, "\%s \ \ t \ \ t\%s \ \ t\%
   d \setminus t\%d \setminus t \setminus t\%s \setminus 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, "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 \setminus t\%s \setminus t\%d \setminus n",
      rejectedProcesses[i].
      orderNumber, 'A' +
      rejectedProcesses[i].
      products , intToTime(
      rejectedProcesses[i].
      dueDate), rejectedProcesses
      [i]. quantity);
fprintf(file, "\t - END - \t );
fprintf(file , "\n");
-----
   _____
   \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]++;
```



8

```
fprintf(file, "\tNumber of
                ToTalproducedQuantity
                    [i] += day[i][j]
                                                     products produced:\t\t %d (
                                                     in total) \n",
                    ].
                    producedQuantity
                                                     ToTalproducedQuantity[i]);
                    ; // Increment
                                                  int total = endPeiod * (300 +
                    To Talproduced Quantity
                                                     100 * i);
                                                  float Utilization =
              }
                                                     ToTalproducedQuantity[i] *
          }
                                                     100 / total;
                                                  fprintf(file, "\tUtilization
        writeToPipe(pipe_fd[1],
                                                     of the plant: \t\t %.1f \%\
           usingdays,
                                                     n\n", Utilization);
           ToTalproducedQuantity);
            // Write usingdays and
                                                  ALLToTalproducedQuantity =
                                                     ALLToTalproducedQuantity +
            ToTalproducedQuantity
           to the pipe
                                                     ToTalproducedQuantity[i];
        fflush (file);
                                                  AllTotal += total;
        exit(0);
                                                }
      }
                                              float Utilization =
    else
                                                 ALLToTalproducedQuantity * 100
                                                 / AllTotal;
      {
                                              fprintf(file, "Overall of
        waitpid(child_pid[i], NULL
           , 0);
                                                 utilization: \t\t\t %.1f \%\n"
                                                 , Utilization);
      }
  }
                                              fflush (file);
                                             return;
// Parent process
waitpid(prev_pid, NULL, 0);
                                       }
                                       F. RUNPLS.C
// Read usingdays and
   ToTalproducedQuantity from each
                                       #include "runpls.h"
    child process
fprintf(file, "\n
                                       // seperate the alg from here
    - End -
                                       // after decided which alg you are going
                                            to use, write it in readme
     \n \n \];
                                       // read the process from 'processes'(
fprintf(file, "=========
                                           globle)
                                       // and write the result into 'day'(
                                           globle)
   \n");
                                       //#define _DEBUG_ // to debug uncomment
fprintf(file, "\n%s\n\n", "***
                                           this line and run 'gcc runpls.c'
   PERFORMANCE");
                                       #ifdef _DEBUG_
int ALLToTalproducedQuantity = 0;
                                       Process processes [10000];
int AllTotal = 0;
                                       int processesCount;
                                       DayArrange day[3][10000];
for (int i = 0; i < 3; i++)
                                       int dayCount[3];
                                       time_t startPeiod;
                                       int endPeiod;
    fprintf(file, "Plant %c:\n",
       X' + i);
                                       int main(){
    readFromPipe(pipe_fd[0],
                                         // set process...
       usingdays,
                                         // call algrothm
       ToTalproducedQuantity);
                                         // print some debug output
    fprintf(file, "\tNumber of
                                         initTime("2022-01-01"); //
       days in use:\t\t\t %d days\
                                         endPeiod = timeToInt("2022-01-30");
       n", usingdays[i]);
                                         processes[processesCount ++] = (
```

```
Process) {"P1000", 3, 1000, 0, 1};
                                                  Process) {"P0011", 8, 3, 0, 0};
   //B (3)
                                              // runPLS(1);
processes[processesCount ++] = (
                                              int m;
   Process) {"P1001", 3, 700, 2, 1};
processes[processesCount ++] = (
                                            #endif
   Process) {"P1002", 3, 1200, 1, 2};
processes[processesCount ++] = (
   Process) {"P1003", 6, 1300, 2, 0};
                                            void FCFS(){
processes[processesCount ++] = (
                                              int i, j, k;
   Process) {"P1004", 3, 1400, 1, 2};
                                              int avaliableDays=endPeiod;
processes[processesCount ++] = (
                                              int XDays=endPeiod;
   Process) {"P1005", 3, 1500, 1, 0};
                                              int YDays=endPeiod;
processes[processesCount ++] = (
                                              int ZDays=endPeiod;
   Process) {"P1006", 4, 2000, 0, 1};
                                              int currentDay=0;
processes[processesCount ++] = (
   Process) {"P1007", 4, 2200, 2, 1};
                                              int XYZStatus [3] = \{0, 0, 0, 0\};
processes[processesCount ++] = (
                                              for(i=0; i < processesCount; i++)
   Process) {"P1008", 4, 2400, 2, 2};
                                                int productivity=0;
processes[processesCount ++] = (
                                                loop:
   Process) {"P1009", 4, 2600, 1, 0};
                                                for (k=0; k<3; k++)
processes[processesCount ++] = (
                                                  while (XYZStatus [0]!=0&&XYZStatus
   Process) {"P1010", 4, 2800, 0, 2};
                                                      [1]! = 0 \& XYZStatus [2]! = 0)
processes[processesCount ++] = (
                                                     XYZStatus[0]--;
   Process) {"P1011", 6, 3000, 0, 2};
                                                     XYZStatus[1]--;
runPLS(0);
                                                     XYZStatus[2]--;
int i, j;
                                                     currentDay++;
// for (m=0; m < dayCount; m++)
                                                  if (XYZStatus[k]==0){
//
     day[]. Product. orderNumber,
//
     day[m]. Product. dueDate,
                                                     if(k==0){
//
     day[m]. Product. quantity,
     day[m]. Product. categorie,
                                                       productivity=300;
//
//
     day[m]. Product. accepted);
                                                       int needDays=processes[i].
// }
                                                          quantity/productivity;
processes[processesCount ++] = (
                                                       int jugde=processes[i].
   Process) {"P0000", 3, 1000, 0, 1};
                                                          quantity%productivity;
processes[processesCount ++] = (
                                                       if (jugde!=0) {
   Process) {"P0001", 4, 200, 2, 1};
                                                         needDays++;
processes[processesCount ++] = (
   Process) {"P0002", 5, 300, 1, 2};
                                                       if (needDays>processes[i].
processes[processesCount ++] = (
                                                          dueDate-currentDay){
   Process) {"P0003", 6, 400, 2, 0};
                                                         processes[i].accepted=0;
processes[processesCount ++] = (
                                                         continue;
   Process) {"P0004", 7, 1400, 1, 2};
processes[processesCount ++] = (
                                                       else if (needDays <= processes [ i</pre>
   Process) {"P0005", 2, 2400, 1, 0};
                                                          ]. dueDate-currentDay&&
processes[processesCount ++] = (
                                                          needDays <= XDays) {
   Process) {"P0006", 3, 500, 0, 1};
                                                         XYZStatus[k]=needDays;
processes[processesCount ++] = (
                                                         XDays—=needDays;
   Process) {"P0007", 1, 600, 2, 1};
                                                         processes [i]. accepted=1;
processes[processesCount ++] = (
                                                         for (j=0; j < needDays; j++){
   Process) {"P0008", 5, 900, 2, 2};
processes[processesCount ++] = (
                                                           day [k] [dayCount[k]].
   Process) {"P0009", 6, 2000, 1, 0};
                                                              Product=processes[i];
processes[processesCount ++] = (
                                                           day[k][dayCount[k]].
   Process) {"P0010", 7, 1230, 0, 2};
                                                               producedQuantity=
processes[processesCount ++] = (
                                                               productivity;
```



```
if (jugde && j = \text{needDays} - 1)
                                               else if (k==2)
        day[k][dayCount[k]].
                                                 productivity=500;
            producedQuantity=
                                                 int needDays=processes[i].
                                                     quantity/productivity;
            jugde;
                                                 int jugde=processes[i].
      dayCount[k]++;
                                                     quantity%productivity;
    }
                                                 if (jugde!=0) {
                                                   needDays++;
    break:
                                                 if (needDays>processes[i].
                                                     dueDate-currentDay){
  }else{
    processes[i].accepted=0;
                                                    processes[i].accepted=0;
                                                   continue;
else\ if(k==1)
                                                 else if (needDays <= processes [i
                                                     ]. dueDate-currentDay&&
  productivity=400;
                                                     needDays <= ZDays) {
  int needDays=processes[i].
                                                   XYZStatus [k]=needDays;
     quantity/productivity;
                                                   ZDays-=needDays;
  int jugde=processes[i].
                                                    processes[i].accepted=1;
     quantity%productivity;
                                                   for (j=0; j < needDays; j++)
  if (jugde!=0) {
                                                      day[k][dayCount[k]].
    needDays++;
                                                         Product=processes[i];
                                                      day[k][dayCount[k]].
  if (needDays>processes[i].
                                                         producedQuantity=
     dueDate-currentDay){
                                                         productivity;
    processes [i]. accepted=0;
                                                      if (jugde && j==needDays-1)
    continue;
                                                        day[k][dayCount[k]].
  if (needDays <= processes [i].
                                                           producedQuantity=
     dueDate-currentDay&&
                                                           jugde;
     needDays <= YDays) {
    XYZStatus[k]=needDays;
                                                      dayCount[k]++;
    YDays—=needDays;
    processes[i].accepted=1;
                                                   break;
    for(j=0; j < needDays; j++)
                                                 } else {
      day[k][dayCount[k]].
          Product=processes[i];
                                                    processes[i].accepted=0;
      day[k][dayCount[k]].
          producedQuantity=
          productivity;
      if (jugde && j==needDays-1)
        day[k][dayCount[k]].
                                           }
            producedQuantity=
            jugde;
                                         }
                                      }
      dayCount[k]++;
    break;
                                      void priorityScheduling() {
                                         int dayCounting = 0;
  } else {
    processes[i].accepted=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 (iugde!=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 (jugde && j==needDays-1)
  if (rawDay[i].dueDate-
     currentDay >= needDays&&
     needDays <= YDays) {
                                                       day[k][dayCount[k]].
    XYZStatus[k]=needDays;
                                                           producedQuantity=
    YDays-eneedDays;
                                                           jugde;
    processes [acceptedIndex[i]].
       accepted=1;
                                                     dayCount[k]++;
    for (j=0; j < needDays; j++)
      day[k][dayCount[k]].
                                                   break;
          Product=rawDay[i];
      day[k][dayCount[k]].
                                                 } else {
          producedQuantity=
                                                   processes [acceptedIndex[i]].
          productivity;
                                                       accepted=0;
      if (jugde && j==needDays-1)
        day[k][dayCount[k]].
            producedQuantity=
                                            }
            jugde;
                                          }
                                        }
      dayCount[k]++;
                                      }
    break;
                                      void CDF() {
  } else {
                                        int* mark = (int*) malloc(size of (int)*
                                            processesCount);
    processes [acceptedIndex[i]].
                                        int dayCounting = 0;
       accepted=0;
                                        Process* rawDay = (Process*) malloc(
  }
                                            sizeof(Process)*processesCount);
else\ if(k==2)
                                        int* acceptedIndex = (int*) malloc(
                                            sizeof(int)*processesCount);
  productivity=500;
                                        int i;
  int needDays=rawDay[i].
                                        memset(mark, 0, sizeof(int)*
     quantity/productivity;
                                            processesCount);
  int jugde=rawDay[i].quantity%
                                        for (i = 0; i < processesCount; i++) {
     productivity;
                                          int min = 0 x 7 ffffffff;
                                          int minIndex = -1;
  if (jugde!=0) {
    needDays++;
                                          for (int j = 0; j < processesCount;
                                              i++) {
  if (needDays>rawDay[i].dueDate-
                                            if (mark[j] == 0 \&\& processes[j].
     currentDay){
                                                dueDate < min) {
    processes [acceptedIndex[i]].
                                               min = processes[j].dueDate;
       accepted=0;
                                               minIndex = j;
  else if (rawDay[i].dueDate-
     currentDay >= needDays&&
                                          mark[minIndex] = 1;
     needDays <= ZDays) {
                                          acceptedIndex[dayCounting] =
    XYZStatus[k]=needDays;
                                              minIndex;
    ZDays-needDays;
                                          rawDay[dayCounting++] = processes[
    processes [acceptedIndex[i]].
                                              minIndex];
       accepted=1;
                                        }
    for(j=0; j < needDays; j++)
      day[k][dayCount[k]].
                                        int j,k;
          Product=rawDay[i];
                                        int avaliableDays=endPeiod;
      day[k][dayCount[k]].
                                        int XDays=endPeiod;
          producedQuantity=
                                        int YDays=endPeiod;
          productivity;
                                        int ZDays=endPeiod;
```



```
int currentDay=0;
                                                              accepted=0;
int XYZStatus [3] = \{0,0,0,0\};
for(i=0; i < processesCount; i++){
                                                      else if(k==1)
  int productivity=0;
  for (k=0; k<3; k++)
                                                        productivity=400;
                                                        int needDays=rawDay[i].
    while (XYZStatus [0]!=0&&XYZStatus
        [1]!=0\&\&XYZStatus[2]!=0){
                                                           quantity/productivity;
      XYZStatus[0]--;
                                                        int jugde=rawDay[i].quantity%
      XYZStatus[1]--;
                                                           productivity;
      XYZStatus[2]--;
                                                        if (jugde!=0) {
                                                          needDays++;
      currentDay++;
    if (XYZStatus[k]==0){
                                                        if (needDays>rawDay[i].dueDate-
      if(k==0)
                                                           currentDay){
                                                          processes[acceptedIndex[i]].
        productivity=300;
                                                              accepted=0;
        int needDays=rawDay[i].
            quantity/productivity;
                                                        if (rawDay[i].dueDate-
        int jugde=rawDay[i].quantity%
                                                           currentDay >= needDays&&
            productivity;
                                                           needDays <= YDays) {
        if (jugde!=0) {
                                                          XYZStatus[k]=needDays;
          needDays++;
                                                          YDays—=needDays;
                                                          processes [acceptedIndex[i]].
        if (needDays>rawDay[i].dueDate-
                                                              accepted=1;
            currentDay){
                                                          for(j=0; j < needDays; j++)
           processes [acceptedIndex[i]].
                                                            day[k][dayCount[k]].
              accepted=0;
                                                                Product=rawDay[i];
                                                            day[k][dayCount[k]].
        else if (rawDay[i].dueDate-
                                                                producedQuantity=
            currentDay >= needDays&&
                                                                productivity;
            needDays <= XDays) {
                                                            if (jugde && j==needDays -1)
          XYZStatus[k]=needDays;
          XDays—=needDays;
                                                              day[k][dayCount[k]].
          processes [acceptedIndex[i]].
                                                                  producedQuantity=
              accepted=1;
                                                                  jugde;
          for(j=0; j < needDays; j++)
                                                            dayCount[k]++;
             day[k][dayCount[k]].
                Product=rawDay[i];
                                                          break;
             day[k][dayCount[k]].
                producedQuantity=
                                                        } else {
                                                          processes [acceptedIndex[i]].
                productivity;
             if (jugde && j==needDays-1)
                                                              accepted=0;
               day[k][dayCount[k]].
                  producedQuantity=
                                                      else if (k==2)
                  jugde;
                                                        productivity=500;
             dayCount[k]++;
                                                        int needDays=rawDay[i].
           }
                                                           quantity/productivity;
                                                        int jugde=rawDay[i].quantity%
                                                           productivity;
          break;
                                                        if (jugde!=0) {
        } else {
                                                          needDays++;
           processes [acceptedIndex[i]].
```



```
if (needDays>rawDay[i].dueDate-
                                              #include "tools.h"
              currentDay){
             processes [acceptedIndex[i]].
                                              void runPLS(int alg);
                accepted=0;
                                              // void algPR (...)
                                              // void algFCFS(...)
          else if (rawDay[i].dueDate-
              currentDay >= needDays&&
                                              // void algSJF (...)
              needDays <= ZDays) {
                                              H. TOOLS.C
             XYZStatus[k]=needDays;
            ZDays-needDays;
                                              #include "tools.h"
             processes [acceptedIndex[i]].
                accepted=1;
                                              //#define _DEBUG_
            for (j=0; j < needDays; j++)
                                              #ifdef _DEBUG_
               day[k][dayCount[k]].
                                              Process processes [10000];
                  Product=rawDay[i];
                                              DayArrange day[10000];
              day[k][dayCount[k]].
                                              int endPeiod;
                  producedQuantity=
                                              time t startPeiod;
                  productivity;
               if (jugde && j==needDays-1)
                                              int main(){
                 day[k][dayCount[k]].
                                              #endif
                    producedQuantity=
                    jugde;
                                              int streq(const char* a, const char* b)
                                                 if (strlen(a) != strlen(b)) return 0;
               dayCount[k]++;
                                                 return !memcmp(a, b, sizeof(char) *
            break;
                                                    strlen(b));
          } else {
             processes [acceptedIndex[i]].
                                              char** genCommand(char* str , int* len) {
                                                 char** result = malloc(10*sizeof(char
                accepted=0;
          }
                                                    *));
                                                 int 1 = 0;
        }
      }
                                                 if (str[strlen(str) - 1] == '\n'
                                                     str[strlen(str) - 1] == '\r'
    }
                                                   str[strlen(str) - 1] = 0; // remove
  }
                                                      the \ \ r
}
                                                 char *token = strtok(str, " ");
void runPLS(int alg) {
                                                 while( token != NULL ) {
  switch (alg) {
                                                   result[1++] = token;
                                                   token = strtok(NULL, " ");
                                                 }
  case 0:
                                                *len = 1;
    FCFS();
                                                 return result;
    break;
                                              }
  case 1:
    priorityScheduling();
                                              int checkCommand(char* str) {
    break;
                                                 if (streq(str, "addPEIOD"))
  case 2:
                                                   return 0;
    CDF();
                                                 if (streq(str, "addORDER"))
    break;
                                                   return 1;
                                                 if (streq(str, "addBATCH"))
  }
                                                   return 2;
                                                 if (streq(str, "runPLS"))
G. RUNPLS.H
                                                   return 3:
```

```
void errorUsage(int c) {
  if (streq(str, "exitPLS"))
    return 4;
                                                switch (c) {
  if (streq(str, "addPERIOD"))
                                                case 0:
    return 0:
                                                  printf("Usage: addPERIOD start date
                                                     end_date\n");
  return -1;
                                                  printf("specify the period for
}
                                                     scheduling the production \n");
int commandAlg(char* alg) {
                                                case 1:
  if (streq(alg, "FCFS"))
                                                  printf("Usage: addORDER order number
    return 0;
                                                      due_date quantity product_name \n
  if (streq(alg, "PR"))
    return 1;
                                                  printf("add an order and the details
  if (streq(alg, "CDF"))
                                                      to the scheduler.\n");
    return 2;
  return -1;
                                                  printf("Usage: addBATCH filename\n")
                                                  printf("input multiple orders in one
void initTime(char* startTime) {
                                                      batch file.\n");
  struct tm tm;
                                                case 3:
  memset(&tm, 0, sizeof(tm));
                                                  printf("Usage: runPLS algorithm |
  strptime (startTime, "%Y-%m-%d", &tm);
                                                     printREPORT [> filename ]\n");
  startPeiod = mktime(&tm);
                                                  printf("generate a schedule with the
                                                      specified algorithm.\n");
                                                }
int timeToInt(char* str) {
                                              }
  struct tm tm;
  memset(&tm, 0, sizeof(tm));
                                              int checkRunUsage (char** c, int 1) {
  strptime (str, "%Y-%m-%d", &tm);
                                                if (1 < 4) return 1 == 2;
  return (int)(difftime(mktime(&tm),
                                                if (!streq(c[2], "|") || !streq(c[3],
     startPeiod)/86400 + 0.5);
                                                   "printREPORT")) return 0;
}
                                                if (1 < 6) return 1 == 4;
                                                if (!streq(c[4], ">")) return 0;
char* intToTime(int i) {
                                                return 1 == 6;
  char* str = malloc(20*sizeof(char));
  struct tm tm = *(localtime(&startPeiod
                                              I. TOOLS.H
     ));
  tm.tm_mday += i;
                                              #ifndef TOOLS H
                                              #define TOOLS H
  mktime(&tm);
  strftime (str, 20, "%Y-%m-%d", &tm);
  return str;
                                              #define XOPEN SOURCE
}
                                              #include < stdlib .h>
                                              #include < stdio.h>
void printMenu(){
  printf("Please enter:\n> ");
                                              #include <time.h>
                                              #include < string . h>
void errorAlg(char* str) {
                                              typedef struct proc {
  printf("runPLS: %s: algorithm not
                                                char orderNumber[100];
     found \n", str);
                                                int dueDate;
}
                                                int quantity;
                                                int products;
void errorCommand(char* str) {
                                                int categorie;
  printf("%s: command not found\n", str)
                                                int accepted; // modify by runpls
                                                int plantX; // the days of each plant
}
                                                   used
                                                int plantY;
```

## IEEE Access

<pre>int plantZ;</pre>	P0055	2024-06-01	
<pre>Process;</pre>		-04 4	1200
		PLANT_X	
Annodof oAnnoA donAnnon (		2024-06-05	100
<pre>typedef struct dayArrange {   Process Product;</pre>		-06 2	400
int producedQuantity;		PLANT_X 2024-06-07	
DayArrange;		-11 5	1500
, 2 ujge ,		PLANT_X	1000
extern Process processes [10000];		2024 - 06 - 12	
extern int processesCount;	2024 - 06	-16 5	1300
<pre>extern DayArrange day[3][10000];</pre>		PLANT_X	
extern int dayCount[3];		2024-06-17	
extern time_t startPeiod;		-18 2	400
extern int endPeiod;		PLANT_X 2024-06-19	
		-28 10	3000
<pre>int streq(const char* a, const char* b);</pre>		PLANT_X	3000
int stred (court chart a, court chart s),		2024-06-29	
<pre>void printMenu();</pre>		-29 1	300
<pre>void errorCommand(char* str);</pre>		PLANT_X	
<pre>void errorAlg(char* str);</pre>		2024 - 06 - 01	
<pre>void errorUsage(int c);</pre>		-03 3	1000
		PLANT_Y	
<pre>char** genCommand(char* str , int* len);</pre>		2024-06-04	000
int sheek Command (about str)		-06 3	900
<pre>int checkCommand(char* str); int commandAlg(char* str);</pre>		PLANT_Y 2024-06-07	
int checkRunUsage(char** command, int		-12 6	2400
len);		PLANT_Y	2.00
. , ,		2024-06-13	
// the int is the day since startpeiod		-19 7	2800
// startpeiod is 0, and so on		PLANT_Y	
// you can call initTime to set		2024 - 06 - 20	
startpeiod		-24 5	1900
<pre>void initTime(char* startTime);</pre>	D0002	PLANT_Y	
<pre>int timeToInt(char* str); char* intToTime(int i);</pre>		2024-06-25 $-27$ 3	1100
char* introffme (int 1),	2024-00	PLANT_Y	1100
#endif	P0053	2024-06-28	
	2024-06		200
APPENDIX B		PLANT_Y	
SAMPLE OUTPUTS	P0063	2024 - 06 - 29	
A. REPORT_CDF.TXT	2024 - 06		400
	T-0.1.0.0	PLANT_Y	
***PLS Schedule Analysis Report***	P0100	2024-06-01	500
Algorithm used CDE	2024-06		500
Algorithm used: CDF	P0069	PLANT_Z 2024-06-02	
There are 24 Orders ACCEPTED. Details	2024-06		1100
are as follows:	2021 00	PLANT_Z	1100
ORDER NUMBER START END	P0015	2024-06-05	
DAYS QUANTITY PLANT	2024-06	-06 2	700
=======================================		PLANT_Z	
=======================================	P0095	2024 - 06 - 07	
====	2024 - 06	-14 8	3600



PLANT_Z	P0021	
P0024 2024-06-15	2024 - 09 - 09	
2024 - 06 - 16 2 600	P0023	Product_A
PLANT_Z	2024 - 07 - 30	4700
P0017 2024-06-17	P0025	Product_B
2024-06-17 1 500	2024 - 08 - 02	4600
PLANT_Z	P0026	Product_E
P0091 2024-06-18	2024 - 07 - 24	900
2024 - 06 - 27 $10$ $4700$	P0027	Product_C
PLANT_Z	2024 - 09 - 09	1600
P0081 2024-06-28	P0028	Product_A
2024-06-28 1 300	2024 - 08 - 12	2200
PLANT_Z	P0029	Product_F
P0002	2024 - 08 - 23	
2024-06-29 1 500		Product_F
PLANT_Z	2024 - 07 - 12	
- END -		Product_C
	2024-07-29	
		Product_F
	2024-09-27	
====	P0033	
There are 76 Orders REJECTED. Details	2024-08-23	
are as follows:	P0034	
	2024-06-21	
QUANTITY Due Date	P0035	
QOANIII I	2024-07-26	
		Product_B
	2024-08-10	
==== P0001		
	D0020	Deadnet C
P0001 Product_E		Product_C
2024-06-25 4800	2024 - 07 - 11	3000
2024-06-25 4800 P0003 Product_A	2024-07-11 P0040	3000 Product_D
2024-06-25 4800 P0003 Product_A 2024-07-10 3900	$\substack{2024-07-11\\ \text{P0040}\\ 2024-09-02}$	3000 Product_D 4600
2024-06-25 4800 P0003 Product_A 2024-07-10 3900 P0004 Product_B	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \end{array}$	3000 Product_D 4600 Product_E
2024-06-25 4800 P0003 Product_A 2024-07-10 3900 P0004 Product_B 2024-08-09 3200	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \end{array}$	3000 Product_D 4600 Product_E 1300
2024-06-25 4800 P0003 Product_A 2024-07-10 3900 P0004 Product_B 2024-08-09 3200 P0005 Product_C	2024-07-11 P0040 2024-09-02 P0041 2024-08-16 P0042	3000 Product_D 4600 Product_E 1300 Product_D
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	$2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22$	3000 Product_D 4600 Product_E 1300 Product_D 200
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	2024-07-11 $P0040$ $2024-09-02$ $P0041$ $2024-08-16$ $P0042$ $2024-08-22$ $P0043$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G
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	2024-07-11 $P0040$ $2024-09-02$ $P0041$ $2024-08-16$ $P0042$ $2024-08-22$ $P0043$ $2024-08-28$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700
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	$2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22 \\ P0043 \\ 2024-08-28 \\ P0044$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D
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	$\begin{array}{c} 2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22 \\ P0043 \\ 2024-08-28 \\ P0044 \\ 2024-07-30 \\ \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300
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	2024-07-11 $P0040$ $2024-09-02$ $P0041$ $2024-08-16$ $P0042$ $2024-08-22$ $P0043$ $2024-08-28$ $P0044$ $2024-07-30$ $P0045$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C
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	2024-07-11 $P0040$ $2024-09-02$ $P0041$ $2024-08-16$ $P0042$ $2024-08-22$ $P0043$ $2024-08-28$ $P0044$ $2024-07-30$ $P0045$ $2024-09-01$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600
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	2024-07-11 $P0040$ $2024-09-02$ $P0041$ $2024-08-16$ $P0042$ $2024-08-22$ $P0043$ $2024-08-28$ $P0044$ $2024-07-30$ $P0045$ $2024-09-01$ $P0046$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A
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	$\begin{array}{c} 2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22 \\ P0043 \\ 2024-08-28 \\ P0044 \\ 2024-07-30 \\ P0045 \\ 2024-09-01 \\ P0046 \\ 2024-09-02 \\ \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000
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	$\begin{array}{c} 2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22 \\ P0043 \\ 2024-08-28 \\ P0044 \\ 2024-07-30 \\ P0045 \\ 2024-09-01 \\ P0046 \\ 2024-09-02 \\ P0047 \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B
2024-06-25	$\begin{array}{c} 2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22 \\ P0043 \\ 2024-08-28 \\ P0044 \\ 2024-07-30 \\ P0045 \\ 2024-09-01 \\ P0046 \\ 2024-09-02 \\ P0047 \\ 2024-07-21 \\ \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500
2024-06-25	$\begin{array}{c} 2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22 \\ P0043 \\ 2024-08-28 \\ P0044 \\ 2024-07-30 \\ P0045 \\ 2024-09-01 \\ P0046 \\ 2024-09-02 \\ P0047 \\ 2024-07-21 \\ P0048 \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D
2024-06-25	$\begin{array}{c} 2024-07-11 \\ P0040 \\ 2024-09-02 \\ P0041 \\ 2024-08-16 \\ P0042 \\ 2024-08-22 \\ P0043 \\ 2024-08-28 \\ P0044 \\ 2024-07-30 \\ P0045 \\ 2024-09-01 \\ P0046 \\ 2024-09-02 \\ P0047 \\ 2024-07-21 \\ P0048 \\ 2024-06-22 \\ \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100
2024-06-25	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I
2024-06-25	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \\ 2024-07-21 \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I 2600
2024-06-25	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \\ 2024-07-21 \\ \text{P0050} \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I 2600 Product_F
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	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \\ 2024-07-21 \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I 2600
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	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \\ 2024-07-21 \\ \text{P0050} \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I 2600 Product_F 800 Product_H
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	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \\ 2024-07-21 \\ \text{P0050} \\ 2024-08-09 \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I 2600 Product_F 800
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	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \\ 2024-07-21 \\ \text{P0050} \\ 2024-08-09 \\ \text{P0051} \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I 2600 Product_F 800 Product_H
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	$\begin{array}{c} 2024-07-11 \\ \text{P0040} \\ 2024-09-02 \\ \text{P0041} \\ 2024-08-16 \\ \text{P0042} \\ 2024-08-22 \\ \text{P0043} \\ 2024-08-28 \\ \text{P0044} \\ 2024-07-30 \\ \text{P0045} \\ 2024-09-01 \\ \text{P0046} \\ 2024-09-02 \\ \text{P0047} \\ 2024-07-21 \\ \text{P0048} \\ 2024-06-22 \\ \text{P0049} \\ 2024-07-21 \\ \text{P0050} \\ 2024-08-09 \\ \text{P0051} \\ 2024-06-07 \end{array}$	3000 Product_D 4600 Product_E 1300 Product_D 200 Product_G 4700 Product_D 2300 Product_C 1600 Product_A 2000 Product_B 4500 Product_D 3100 Product_I 2600 Product_F 800 Product_H 1700

# IEEE Access

P0054	Product_F	P0089	Product_I	
2024-08-30	2600	2024-06-13	3300	
P0056	Product_C	P0093	Product_A	
	100	2024-06-06	2500	
P0057	Product_F	P0096	Product_I	
2024 - 08 - 05	4800	2024 - 07 - 23	1100	
P0058	Product_A	P0097	Product_C	
2024-06-21	4800	2024-07-18	1900	
P0059	Product_I	P0098	Product_D	
2024-06-11	2500		1200	
P0060	Product_E	P0099		
2024-08-27	3500	2024-06-18	4200	
P0061	Product_C	– END –		
2024-06-18	3800			
P0062	Product_D			
2024-07-25	3400			
P0064	Product_D	====		
2024-06-06	3900	==== Plant X		
		<del>-</del>	D	01
P0065	Product_B		Product Name	Order
	3300	_	uantity (Produced)	
P0066	Product_D	DueDate	D 1 . II	D0055
	1700		Product_H	P0055
P0067	Product_F	300		
	2200	2024-06-05	D 1	D0055
P0070	Product_B	2024-06-02	Product_H	P0055
2024-09-04	800	300		
P0071	Product_A	2024-06-05		
2024-07-17	2600	2024-06-03	Product_H	P0055
P0072	Product_C	300		
2024 - 07 - 12	4000	2024 - 06 - 05		
P0073	Product_G	2024 - 06 - 04	Product_H	P0055
2024 - 07 - 26	2200	300		
P0074	Product_I	2024 - 06 - 05		
	2700	2024 - 06 - 05	Product_D	P0019
P0075	Product_E	300		
	600	2024 - 06 - 09		
P0076	Product_B	2024 - 06 - 06	Product_D	P0019
2024 - 07 - 29	3300	100		
P0077	Product_I	2024 - 06 - 09		
2024 - 07 - 17	2600	2024 - 06 - 07	Product_B	P0090
P0078	Product_B	300		
2024 - 08 - 31	4400	2024 - 06 - 16		
P0079	Product_D	2024 - 06 - 08	Product_B	P0090
2024 - 08 - 21	4700	300		
P0080	Product_G	2024 - 06 - 16		
2024 - 08 - 10	3400	2024 - 06 - 09	Product_B	P0090
P0082	Product_I	300		
2024 - 07 - 05	5000	2024 - 06 - 16		
P0084	Product_G	2024 - 06 - 10	Product_B	P0090
2024 - 07 - 12	1600	300		
P0086	Product_A	2024 - 06 - 16		
2024 - 08 - 11	3000	2024 - 06 - 11	Product_B	P0090
P0087	Product_G	300		
2024 - 08 - 20	3000	2024 - 06 - 16		
P0088	Product_I	2024 - 06 - 12	Product_C	P0037
2024 - 08 - 27	1300	300		



2024-06-19 2024-06-13	Product C	P0037	2024-06-01	Product_C	P0094
$\begin{array}{c} 300 \\ 2024 - 06 - 19 \end{array}$			$\begin{array}{c} 2024 - 06 - 05 \\ 2024 - 06 - 02 \end{array}$	Product_C	P0094
2024-06-14 300 2024-06-19	Product_C	P0037	$ \begin{array}{r} 400 \\ 2024 - 06 - 05 \\ 2024 - 06 - 03 \end{array} $	Product C	P0094
2024-06-15	Product_C	P0037	200 $2024-06-05$		
2024 - 06 - 19 $2024 - 06 - 16$ $100$	Product_C	P0037	2024 - 06 - 04 $400$ $2024 - 06 - 08$	Product_H	P0022
$\begin{array}{c} 2024 - 06 - 19 \\ 2024 - 06 - 17 \end{array}$	Product_A	P0006	2024-06-05 400	Product_H	P0022
$   \begin{array}{r}     300 \\     2024 - 06 - 28 \\     2024 - 06 - 18   \end{array} $	Product A	P0006	$\begin{array}{c} 2024 - 06 - 08 \\ 2024 - 06 - 06 \\ 100 \end{array}$	Product_H	P0022
$\begin{array}{c} 100 \\ 2024 - 06 - 28 \end{array}$			$\begin{array}{c} 2024 - 06 - 08 \\ 2024 - 06 - 07 \end{array}$	Product_F	P0036
2024-06-19 300 2024-06-30	Product_F	P0092	400 2024-06-14 2024-06-08	Product F	P0036
2024-06-20	Product_F	P0092	400 2024-06-14	Post 1 of E	D0026
$\begin{array}{c} 2024 - 06 - 30 \\ 2024 - 06 - 21 \\ 300 \end{array}$	Product_F	P0092	2024-06-09 400 2024-06-14	Product_F	P0036
$\begin{array}{c} 2024 - 06 - 30 \\ 2024 - 06 - 22 \\ 300 \end{array}$	Product_F	P0092	2024-06-10 400 2024-06-14	Product_F	P0036
$\begin{array}{c} 2024 - 06 - 30 \\ 2024 - 06 - 23 \end{array}$	Product_F	P0092	2024-06-11 400	Product_F	P0036
$   \begin{array}{r}     300 \\     2024 - 06 - 30 \\     2024 - 06 - 24   \end{array} $	Product F	P0092	$\begin{array}{c} 2024 - 06 - 14 \\ 2024 - 06 - 12 \\ 400 \end{array}$	Product_F	P0036
300 $2024-06-30$			2024-06-14 2024-06-13	Product_D	P0008
2024-06-25 300 2024-06-30	Product_F	P0092	$ \begin{array}{r} 400 \\ 2024 - 06 - 20 \\ 2024 - 06 - 14 \end{array} $	Product_D	P0008
2024-06-26	Product_F	P0092	$\begin{array}{c} 400 \\ 2024 - 06 - 20 \end{array}$		
2024 - 06 - 30 $2024 - 06 - 27$ $300$	Product_F	P0092	2024-06-15 400 2024-06-20	Product_D	P0008
2024 - 06 - 30 $2024 - 06 - 28$ $300$	Product_F	P0092	2024-06-16 400	Product_D	P0008
2024-06-30 2024-06-29	Product_I	P0085	2024-06-20 2024-06-17 400	Product_D	P0008
300 2024-07-27 Plant_Y			2024 - 06 - 20 $2024 - 06 - 18$ $400$	Product_D	P0008
Date	Product Name uantity (Produced)	Order	2024-06-20 2024-06-19 400	Product_D	P0008



2024-06-20			2024-06-08	Product_C	P0095
2024-06-20 400	Product_I	P0068	$500 \\ 2024-06-16$		
2024-07-05			2024-06-09	Product_C	P0095
2024 - 06 - 21	Product_I	P0068	500		
400			2024-06-16	Dec deset C	D0005
2024-07-05 $2024-06-22$	Product I	P0068	2024-06-10 500	Product_C	P0095
400	110000_1	10000	2024-06-16		
2024-07-05	D 1 T	<b>D</b> 0060	2024-06-11	Product_C	P0095
2024-06-23	Product_I	P0068	$500 \\ 2024-06-16$		
2024 - 07 - 05			2024-06-12	Product_C	P0095
2024 - 06 - 24	$Product_I$	P0068	500		
300 $2024-07-05$			2024 - 06 - 16 $2024 - 06 - 13$	Droduat C	P0095
2024-07-03	Product G	P0083	500	rioduci_C	F0093
400	_		2024 - 06 - 16		
2024-07-06	D 1 C	D0002	2024-06-14	Product_C	P0095
2024-06-26	Product_G	P0083	$\begin{array}{c} 100 \\ 2024 - 06 - 16 \end{array}$		
2024-07-06			2024-06-15	Product_B	P0024
2024 - 06 - 27	Product_G	P0083	500		
300			2024-06-22	Decduar D	D0024
2024-07-06 $2024-06-28$	Product A	P0053	2024-06-16	Product_B	P0024
200			2024 - 06 - 22		
2024-07-07	<b>D</b>	D0062	2024-06-17	Product_B	P0017
2024-06-29	Product_I	P0063	$500 \\ 2024-06-28$		
2024-08-07			2024-06-18	Product_A	P0091
Plant_Z			500		
Date	Product Name uantity (Produced)	Order	2024-06-30 2024-06-19	Product_A	D0001
DueDate			500	Product_A	P0091
2024-06-01	Product_E	P0100	2024 - 06 - 30		
500			2024-06-20	Product_A	P0091
2024-06-02 $2024-06-02$	Product G	P0069	500 $2024-06-30$		
500	110ddct_G	1000)	2024-06-21	Product_A	P0091
2024 - 06 - 06			500		
2024-06-03	Product_G	P0069	$\begin{array}{c} 2024 - 06 - 30 \\ 2024 - 06 - 22 \end{array}$	Product_A	P0091
2024-06-06			500	rioduct_A	10091
2024 - 06 - 04	Product_G	P0069	2024 - 06 - 30		
100			2024-06-23	Product_A	P0091
2024-06-06 $2024-06-05$	Product_B	P0015	500 $2024-06-30$		
500	Troduct_B	10013	2024-06-24	Product_A	P0091
2024-06-10			500		
2024-06-06	Product_B	P0015	2024 - 06 - 30 $2024 - 06 - 25$	Product_A	P0091
2024-06-10			500	I TOUUCI_A	10071
2024 - 06 - 07	Product_C	P0095	2024 - 06 - 30		
500			2024-06-26	Product_A	P0091
2024 - 06 - 16			500		



2024-06-30	
2024-06-30 2024-06-27 Product_A P0091	====
200	P0001 2024-06-01
2024-06-30	2024-06-16 16 4800
2024-06-28 Product_F P0081	PLANT_X
300	$\begin{array}{cccccccccccccccccccccccccccccccccccc$
2024-07-09	2024-06-25 9 2600
2024-06-29 Product_H P0002 500	P0030 2024_06_26
2024-08-17	PLANT_X P0030
	PLANT X
- End $-$	PLANT_X P0042 2024-06-29 2024-06-29 1 200
	2024-06-29 1 200
	PLANT_X
	PLANT_X P0002 2024-06-01 2024-06-02 2 500
====	2024-06-02 2 500
***PERFORMANCE	P0004 2024_06_03
	PLANT_Y P0004 2024-06-03 2024-06-10 8 3200
Plant X:	PLANT_Y
Number of days in use:	PLANT_Y P0006 2024-06-11 2024-06-11 1 400
29 days	2024-06-11 1 400
Number of products produced:	PLANT_Y
8100 (in total)	PLANT_Y P0008
Utilization of the plant: 93.0 %	2024-06-18 7 2800 PLANT V
93.0 %	P0016 2024-06-19
Plant Y:	PLANT_Y P0016 2024-06-19 2024-06-25 7 2600
Number of days in use:	PLANT_Y
29 days	PLANT_Y P0027 2024-06-26 2024-06-29 4 1600
Number of products produced:	2024-06-29 4 1600
10700 (in total)	PLANT_Y
Utilization of the plant: 92.0 %	PLANT_Y P0003 2024-06-01 2024-06-08 8 3900
92.0 %	PLANT 7.
Plant Z:	PLANT_Z P0005 2024-06-09
Number of days in use:	2024-06-10 2 700
29 days	PLANT_Z
Number of products produced:	P0007 2024-06-11
12500 (in total)	2024-06-11 1 400
Utilization of the plant: 86.0 %	PLANT_Z P0009 2024-06-12
80.0 %	2024-06-17 6 2600
Overall of utilization:	PLANT_Z
89.0 %	P0013 2024-06-18
B. REPORT_FCFS.TXT	2024-06-19 2 800
	PLANT_Z
***PLS Schedule Analysis Report***	P0017 2024-06-20
Algorithm used: ECES	2024-06-20 1 500
Algorithm used: FCFS	PLANT_Z P0018 2024-06-21
There are 20 Orders ACCEPTED. Details	2024-06-24 4 2000
are as follows:	PLANT_Z
ORDER NUMBER START END	P0026
	2024-06-26 2 $900$
DAYS QUANTITY PLANT	2024-06-26 2 900 PLANT_Z

# IEEE Access

D0022	2024 06 27	P0039	Draduat C
2024 06 27	2024-06-27 1 100	2024 07 11	Product_C
2024-06-27	1 100	2024-07-11	3000
PLAN	NT_Z	P0040	Product_D
P0035 2024-06-29	2024-06-28	2024-09-02	4600
2024 - 06 - 29	2 700	P0041	Product_E
PLAN		2024 - 08 - 16	1300
- END $-$		P0039 2024-07-11 P0040 2024-09-02 P0041 2024-08-16 P0043 2024-08-28 == P0044	Product_G
		2024 - 08 - 28	4700
===========		== P0044	Product_D
=========		== $2024-07-30$	2300
====		P0045	Product_C
There are 80 Or	ders REJECTED. Details	2024 - 09 - 01	1600
are as follo		$ \begin{array}{rcl}  &=& P0044 \\  &=& 2024-07-30 \\  & P0045 \\  && 2024-09-01 \\  &P0046 \\  && 2024-09-02 \\  &P0047 \\  &=& 2024-07-21 \\  &=& P0048 \\  &=& 2024-06-22 \\  $	Product_A
ORDER NUMBER	PRODUCT NAME Due Date	te $2024-09-02$	2000
QUA	NTITY	P0047	Product_B
===========	=======================================	= 2024 $-$ 07 $-$ 21	4500
========		== P0048	Product_D
====		2024-06-22 P0049 2024-07-21 P0050 2024-08-09	3100
P0010		P0049	Product_I
2024 - 06 - 09	3400	2024 - 07 - 21	2600
P0012	Product_D	P0050	Product_F
2024 - 06 - 08	3400	2024 - 08 - 09	800
P0014	Product A	P0051 2024-06-07 P0052	Product_H
2024 - 08 - 09	5000	2024 - 06 - 07	1700
P0015	Product_B	P0052	Product I
	700	2024 - 06 - 17	2200
P0019	Product_D	P0053	2200 Product_A
	400	2024 - 07 - 07	200
P0020	Product_B	P0054	Product_F
	5000	2024 - 08 - 30	2600
P0021	Product_C	P0055	Product_H
2024-09-09	3500	2024-06-05	1200
P0022		P0056	Product C
2024-06-08	900	P0056 2024-08-26	100
	Product_A	P0057	Product F
2024-07-30	4700	2024-08-05 P0058	4800
P0024	Product_B	P0058	Product A
2024-06-22	600	2024-06-21	4800
P0025	Product_B	P0059	Product_I
2024-08-02	4600	2024-06-11	2500
P0028	Product_A	P0060	Product_E
2024-08-12	2200	2024-08-27	3500
P0029	Product_F	P0061	Product_C
2024-08-23	2000	2024-06-18	3800
P0031	Product_C	P0062	Product_D
2024-07-29	2300	2024-07-25	3400
P0033	Product_A	P0063	Product_I
2024-08-23	3100	2024-08-07	400
P0034	Product_H	P0064	Product_D
2024-06-21	4600	2024-06-06	3900
P0036	Product_F	P0065	Product_B
2024-06-14	2400	2024-07-31	3300
P0037	Product_C	P0066	Product_D
2024-06-19	1300	2024-08-17	1700 1700
P0038	Product_B	P0067	Product_F
2024-08-10	3900	2024-09-12	2200
2024-08-10	3300	2024-09-12	2200



P0068	<del>_</del>	P0096	<del>-</del>	
2024 - 07 - 05	1900	2024 - 07 - 23	1100	
P0069	Product_G	P0097	Product_C	
2024 - 06 - 06	1100	2024 - 07 - 18	1900	
P0070	Product_B	P0098	Product_D	
2024 - 09 - 04	800	2024 - 07 - 31	1200	
P0071	Product_A	P0099	Product_A	
2024 - 07 - 17	2600	2024 - 06 - 18	4200	
P0072	Product C	P0100	Product_E	
2024-07-12	4000	2024-06-02	500	
P0073	Product_G	- END -		
2024-07-26	2200	LIND		
	Product_I			
2024-07-10	2700 D			=======
P0075	Product_E	====		
2024-08-18	600	Plant_X	B 1	0.1
P0076	Product_B		Product Name	Order
2024 - 07 - 29	3300	•	uantity (Produced)	
P0077	Product_I	DueDate		
2024 - 07 - 17	2600	2024 - 06 - 01	Product_E	P0001
P0078	Product_B	300		
2024 - 08 - 31	4400	2024 - 06 - 25		
P0079	Product_D	2024 - 06 - 02	Product_E	P0001
2024 - 08 - 21	4700	300		
P0080	Product_G	2024 - 06 - 25		
2024 - 08 - 10	3400	2024-06-03	Product_E	P0001
P0081	Product_F	300		
2024-07-09	300	2024-06-25		
P0082	Product_I	2024-06-04	Product_E	P0001
2024-07-05	5000	300	110ddct_E	10001
P0083	Product_G	2024-06-25		
2024-07-06	1100	2024-06-05	Product_E	P0001
P0084	Product_G	300	Trouuct_E	1 0001
	1600	2024-06-25		
			D 1 E	D0001
P0085	Product_I	2024-06-06	Product_E	P0001
2024-07-27	300	300		
P0086	Product_A	2024-06-25		
2024-08-11	3000	2024-06-07	Product_E	P0001
P0087	Product_G	300		
2024 - 08 - 20	3000	2024 - 06 - 25		
P0088	Product_I	2024 - 06 - 08	Product_E	P0001
2024 - 08 - 27	1300	300		
P0089	Product_I	2024 - 06 - 25		
2024 - 06 - 13	3300	2024 - 06 - 09	Product_E	P0001
P0090	Product_B	300		
2024 - 06 - 16	1500	2024 - 06 - 25		
P0091	Product_A	2024 - 06 - 10	Product_E	P0001
2024-06-30	4700	300		
P0092	Product_F	2024 - 06 - 25		
2024-06-30	3000	2024-06-11	Product_E	P0001
P0093	Product_A	300	ITOGGCL_L	10001
2024-06-06	2500	2024-06-25		
P0094	Product_C	2024-06-23	Product_E	P0001
	1000	300	110uuct_E	10001
2024-06-05				
P0095	Product_C	2024-06-25	Deadnet E	D0001
2024 - 06 - 16	3600	2024-06-13	Product_E	P0001



3	300			2024 - 08 - 17		
2024 - 06 - 2024 - 06 - 14		Product E	P0001	2024-06-02 100	Product_H	P0002
3	300			2024 - 08 - 17		
2024-06-2 $2024-06-15$		Product F	P0001	2024-06-03	Product_B	P0004
3	300	Troduct_L	10001	2024-08-09		
2024-06-2 $2024-06-16$		Droduot E	P0001	2024-06-04	Product_B	P0004
	300	Flouuct_E	F0001	2024 - 08 - 09		
2024-06-2		D 1	D0011	2024-06-05	Product_B	P0004
2024-06-17	300	Product_G	P0011	400 $2024-08-09$		
2024 - 07 - 2				2024 - 06 - 06	Product_B	P0004
2024-06-18	300	Product_G	P0011	400 $2024-08-09$		
2024-07-2				2024-06-07	Product_B	P0004
2024-06-19	300	Product_G	P0011	400 $2024-08-09$		
2024-07-2				2024-08-09	Product_B	P0004
2024-06-20		Product_G	P0011	400		
2024-07-2	300 20			2024 - 08 - 09 $2024 - 06 - 09$	Product B	P0004
2024 - 06 - 21		Product_G	P0011	400	_	
2024-07-2	300 20			$\begin{array}{c} 2024 - 08 - 09 \\ 2024 - 06 - 10 \end{array}$	Product B	P0004
2024 - 06 - 22		Product_G	P0011	400	110000-2	1000.
2024-07-2	300			2024-08-09 2024-06-11	Product A	P0006
2024-06-23		Product_G	P0011	400	Troubet_A	1 0000
2024-07-2	300			$\begin{array}{c} 2024 - 06 - 28 \\ 2024 - 06 - 12 \end{array}$	Droduat D	P0008
2024-06-24		Product_G	P0011	400	Flouuct_D	F0008
	300			2024-06-20	D 1 . D	<b>D</b> 0000
2024-07-2 $2024-06-25$		Product G	P0011	2024-06-13	Product_D	P0008
2	200	_		2024-06-20		
2024-07-2 $2024-06-26$	20	Product_F	P0030	2024-06-14	Product_D	P0008
3	300	1104401_1	10030	2024 - 06 - 20		
2024-07-1 2024-06-27		Product_F	P0030	2024-06-15	Product_D	P0008
	300	110ddct_1	10030	2024-06-20		
2024-07-1 2024-06-28		Droduot E	P0030	2024-06-16	Product_D	P0008
	100	Product_F	P0030	2024-06-20		
2024-07-1		<b>D</b> 1 . D	D0049	2024-06-17	Product_D	P0008
2024-06-29	200	Product_D	P0042	$400 \\ 2024 - 06 - 20$		
2024 - 08 - 2				2024 - 06 - 18	Product_D	P0008
Plant_Y Date		Product Name	Order	$400 \\ 2024 - 06 - 20$		
Number	Qι	nantity (Produced)	51401	2024 - 06 - 19	Product_D	P0016
DueDate 2024-06-01		Product_H	P0002	400 $2024-08-13$		
	400	110duct_11	1 0002	2024-08-13	Product_D	P0016
24						VOLUME 11, 20



	400			2024 - 07 - 10		
2024 - 08 -				2024 - 06 - 09	Product_C	P0005
2024 - 06 - 21		Product_D	P0016	500		
2024 00	400			2024-07-29	D 1	D0005
2024-08-		Dundant D	D0016	2024-06-10	Product_C	P0005
2024-06-22	400	Product_D	P0016	200 $2024-07-29$		
2024-08-				2024-07-29	Droduct U	P0007
2024-06-23		Product_D	P0016	400	riouuci_ii	10007
2024 00 23	400	Troduct_D	10010	2024-09-02		
2024-08-				2024-06-12	Product I	P0009
2024-06-24		Product D	P0016	500	110000	1000)
	400			2024 - 07 - 17		
2024-08-	-13			2024 - 06 - 13	Product_I	P0009
2024-06-25		Product_D	P0016	500		
	200			$2024\!-\!07\!-\!17$		
2024 - 08 -				2024 - 06 - 14	Product_I	P0009
2024 - 06 - 26		Product_C	P0027	500		
	400			2024 - 07 - 17		
2024-09-				2024 - 06 - 15	Product_I	P0009
2024 - 06 - 27		Product_C	P0027	500		
	400			2024-07-17		
2024-09-		<b>D</b> 1 G	D0005	2024-06-16	Product_I	P0009
2024 - 06 - 28		Product_C	P0027	500		
2024 00	400			2024-07-17	D., . d., I	<b>D</b> 0000
2024-09- 2024-06-29		Draduat C	P0027	2024-06-17 100	Product_1	P0009
2024-00-29	400	rioduct_C	F0027	2024-07-17		
2024-09-				2024-06-18	Product F	P0013
Plant_Z	0)			500	TTOddct_L	10013
Date		Product Name	Order	2024 - 08 - 21		
		antity (Produced)	01401	2024-06-19	Product E	P0013
DueDate		(,		300		
2024-06-01		Product_A	P0003	2024 - 08 - 21		
	500			2024 - 06 - 20	Product_B	P0017
2024 - 07 -	-10			500		
2024 - 06 - 02		Product_A	P0003	2024 - 06 - 28		
	500			2024 - 06 - 21	Product_A	P0018
2024-07-				500		
2024 - 06 - 03		Product_A	P0003	2024-09-20	-	
2024 07	500			2024-06-22	Product_A	P0018
2024-07-		Deadwat A	D0002	500		
2024-06-04	500	Product_A	P0003	2024-09-20	Draduat A	P0018
2024-07-				2024-06-23	Product_A	F0018
2024-07-		Product_A	P0003	2024-09-20		
2024 00 03	500	Troduct_A	1 0003	2024-06-24	Product_A	P0018
2024-07-				500	1104401_71	10010
2024-06-06		Product_A	P0003	2024 - 09 - 20		
	500			2024-06-25	Product_E	P0026
2024-07-				500	_	-
2024 - 06 - 07		Product_A	P0003	2024 - 07 - 24		
	500			2024 - 06 - 26	Product_E	P0026
2024-07-	-10			400		
2024 - 06 - 08		Product_A	P0003	2024 - 07 - 24		
	400			2024 - 06 - 27	Product_F	P0032



100	P0001 2024-06-01	
2024 - 09 - 27		4800
2024-06-28 Product_C P0035	PLANT_X	
500	P0011 2024-06-17	
2024 - 07 - 26	2024-06-25 9	2600
2024-06-29 Product_C P0035	PLANT_X	
200	P0030 2024-06-26	
2024 - 07 - 26	2024-06-28 3	700
	PLANT_X	
- End $-$	P0042 2024-06-29	
	2024-06-29 1	200
=======================================	PLANT_X	
=======================================	P0002 2024-06-01	
====	2024-06-02 2	500
	PLANT_Y	
***PERFORMANCE	P0004	
	2024-06-10 8	3200
Plant X:	PLANT_Y	
Number of days in use:	P0006	
29 days	2024-06-11 1	400
Number of products produced:	PLANT_Y	
8300 (in total)	P0008 2024-06-12	
Utilization of the plant:	2024-06-18 7	2800
95.0 %	PLANT_Y	2000
73.0 %	P0016 2024-06-19	
Plant Y:	2024-06-25 7	2600
Number of days in use:	PLANT_Y	2000
29 days	P0027 2024-06-26	
Number of products produced:	2024-06-29 4	1600
11100 (in total)	PLANT_Y	1000
Utilization of the plant:	P0003 2024-06-01	
95.0 %	2024-06-08 8	3900
73.0 %	PLANT_Z	3700
Plant Z:	P0005 2024-06-09	
Number of days in use:	2024-06-10 2	700
29 days	PLANT_Z	700
Number of products produced:	P0007 2024-06-11	
12600 (in total)	2024-06-11 1	400
Utilization of the plant:	PLANT_Z	400
86.0 %	P0009 2024-06-12	
00.0 //	2024-06-17 6	2600
Overall of utilization:	PLANT_Z	2000
91.0 %	P0013 2024-06-18	
71.0 %	2024-06-19 2	800
C. REPORT_PR.TXT	PLANT_Z	000
***PLS Schedule Analysis Report***	P0017 2024-06-20	
*** LS Schedule Analysis Repolit***	2024-06-20	500
Algorithm used: FCFS	PLANT_Z	300
Algorithm used. PCF3	P0018 2024-06-21	
There are 20 Orders ACCEPTED. Details	2024-06-24 4	2000
are as follows:		2000
ORDER NUMBER START END	PLANT_Z P0026 2024-06-25	
DAYS QUANTITY PLANT	2024 - 06 - 26 $2024 - 06 - 23$	900
	2024-00-20 2 PLANT_Z	200
	P0032 2024-06-27	100
====	2024 - 06 - 27 1	100



27

======================================	2024-06-28 2 700 NT_Z 	2024-08-16 P0043 2024-08-28 P0044 2024-07-30 P0045 2024-09-01 P0046	4600 Product_E 1300 Product_G 4700 Product_D 2300 Product_C 1600 Product_A
QUA	PRODUCT NAME Due Date NTITY	2024-09-02 P0047 2024-07-21	Product_B
=======================================			Product_D
2024 - 06 - 09	Product_H 3400 Product_D	2024 - 07 - 21	Product_I 2600 Product_F
2024-06-08 P0014	3400 Product_A	2024-08-09 P0051	800 Product_H
P0015	5000 Product_B 700	2024-06-07 P0052 2024-06-17	Product_I
P0019 2024-06-09	Product_D 400	P0053 2024-07-07	Product_A 200
P0020 2024-08-26 P0021	Product_B 5000 Product_C	2024-08-30 P0055	Product_F 2600 Product_H
P0022	3500 Product_H 900	P0056	1200 Product_C 100
P0023 2024-07-30	Product_A 4700	P0057 2024-08-05	Product_F 4800
	Product_B 600 Product_B	P0058 2024-06-21 P0059	Product_A 4800 Product_I
2024-08-02 P0028	4600 Product_A 2200	2024-06-11 P0060 2024-08-27	2500 Product_E 3500
2024-08-12 P0029 2024-08-23	Product_F 2000	P0061 2024-06-18	Product_C 3800
P0031 2024-07-29 P0033	Product_C 2300 Product_A	P0062 2024-07-25 P0063	Product_D 3400 Product_I
2024-08-23 P0034	3100 Product_H	2024-08-07 P0064	400 Product_D
2024-06-21 P0036 2024-06-14	4600 Product_F 2400	2024-06-06 P0065 2024-07-31	3900 Product_B 3300
P0037 2024-06-19	Product_C 1300	P0066 2024-08-17	Product_D 1700
P0038 2024-08-10 P0039	Product_B 3900 Product_C	P0067 2024-09-12 P0068	Product_F 2200 Product_I
2024 - 07 - 11	3000	2024 - 07 - 05	1900

# IEEE Access

P0069	Product_G	P0097	Product_C	
2024-06-06	1100	2024-07-18	1900	
P0070	Product_B	P0098	Product_D	
2024-09-04	800	2024-07-31	1200	
P0071		P0099	Product_A	
	Product_A 2600			
2024-07-17		2024-06-18	4200	
P0072	Product_C	P0100	Product_E	
2024-07-12	4000	2024 - 06 - 02	500	
P0073	Product_G	- END $-$	-	
2024-07-26	2200			
P0074	Product_I			======
2024-07-10	2700			=======
P0075	Product_E	====		
2024-08-18	600	Plant_X		
P0076	Product_B		Product Name	Order
2024-07-29	3300	_	uantity (Produced)	
P0077	Product_I	DueDate		
2024 - 07 - 17	2600	2024-06-01	Product_E	P0001
P0078	Product_B	300		
2024 - 08 - 31	4400	2024 - 06 - 25		
P0079	Product_D	2024 - 06 - 02	Product_E	P0001
2024 - 08 - 21	4700	300		
P0080	Product_G	2024 - 06 - 25		
2024 - 08 - 10	3400	2024 - 06 - 03	Product_E	P0001
P0081	Product_F	300		
2024 - 07 - 09	300	2024 - 06 - 25		
P0082	Product_I	2024 - 06 - 04	Product_E	P0001
2024 - 07 - 05	5000	300		
P0083	Product_G	2024 - 06 - 25		
2024 - 07 - 06	1100	2024 - 06 - 05	Product_E	P0001
P0084	Product_G	300		
2024 - 07 - 12	1600	2024 - 06 - 25		
P0085	Product_I	2024 - 06 - 06	Product_E	P0001
2024 - 07 - 27	300	300		
P0086	Product_A	2024 - 06 - 25		
2024 - 08 - 11	3000	2024 - 06 - 07	Product_E	P0001
P0087	Product_G	300		
2024 - 08 - 20	3000	2024 - 06 - 25		
P0088	Product_I	2024 - 06 - 08	Product_E	P0001
2024 - 08 - 27	1300	300		
P0089	Product_I	2024 - 06 - 25		
2024 - 06 - 13	3300	2024 - 06 - 09	Product_E	P0001
P0090	Product_B	300		
2024 - 06 - 16	1500	2024 - 06 - 25		
P0091	Product_A	2024 - 06 - 10	Product_E	P0001
2024 - 06 - 30	4700	300		
P0092	Product_F	2024 - 06 - 25		
2024 - 06 - 30	3000	2024 - 06 - 11	Product_E	P0001
P0093	Product_A	300		
2024 - 06 - 06	2500	2024 - 06 - 25		
P0094	Product_C	2024 - 06 - 12	Product_E	P0001
2024-06-05	1000	300	_	
P0095	Product_C	2024 - 06 - 25		
	3600	2024 - 06 - 13	Product_E	P0001
P0096	Product_I	300	_	
2024 - 07 - 23	1100	2024 - 06 - 25		
-				
20				



2024 - 06 - 14		Product_E	P0001	100		
	300			2024 - 08 - 17		
2024-06-		B 1 - B	D0001	2024-06-03	Product_B	P0004
2024 - 06 - 15		Product_E	P0001	400		
2024-06-	300			2024-08-09	Decduar D	D0004
2024-06-		Product_E	P0001	2024-06-04	Product_B	P0004
2024-00-10	300	rioduct_E	10001	2024-08-09		
2024-06-				2024-06-05	Product_B	P0004
2024-06-17		Product_G	P0011	400	110ddct_B	1000.
	300			2024-08-09		
2024 - 07 -				2024-06-06	Product_B	P0004
2024 - 06 - 18		Product_G	P0011	400		
	300			2024 - 08 - 09		
2024 - 07 -	-20			2024 - 06 - 07	Product_B	P0004
2024 - 06 - 19		Product_G	P0011	400		
	300			2024 - 08 - 09		
2024 - 07 -				2024 - 06 - 08	Product_B	P0004
2024 - 06 - 20		Product_G	P0011	400		
	300			2024-08-09		
2024-07-		<b>D</b> 1	D0011	2024-06-09	Product_B	P0004
2024-06-21		Product_G	P0011	400		
2024 07	300			2024-08-09	D 1 . D	D0004
2024-07-		Duadast C	D0011	2024-06-10	Product_B	P0004
2024-06-22	300	Product_G	P0011	400 $2024-08-09$		
2024-07-				2024-06-11	Product_A	P0006
2024-07-		Product_G	P0011	400	110duct_A	1 0000
2024 00 23	300	TTOUUCI_O	10011	2024-06-28		
2024-07-				2024-06-12	Product_D	P0008
2024-06-24		Product_G	P0011	400	1104401_B	10000
	300	110000	10011	2024-06-20		
2024-07-				2024-06-13	Product_D	P0008
2024 - 06 - 25		Product_G	P0011	400	_	
	200			2024 - 06 - 20		
2024 - 07 -	-20			2024 - 06 - 14	Product_D	P0008
2024 - 06 - 26		Product_F	P0030	400		
	300			2024 - 06 - 20		
2024 - 07 -	-12			2024 - 06 - 15	Product_D	P0008
2024 - 06 - 27		Product_F	P0030	400		
2024 05	300			2024-06-20	D 1 D	<b>D</b> 0000
2024-07-	-12	D 1 4 . E	D0020	2024-06-16	Product_D	P0008
2024 - 06 - 28	100	Product_F	P0030	400		
2024-07-	100			2024-06-20 2024-06-17	Product_D	P0008
2024-07-	-12	Product_D	P0042	400	Product_D	P0008
2024-00-29	200	rioduct_D	10042	2024-06-20		
2024-08-				2024-06-18	Product_D	P0008
Plant_Y	22			400	110ddct_D	1 0000
Date		Product Name	Order	2024-06-20		
Number	Ou	antity (Produced)	01001	2024-06-19	Product_D	P0016
DueDate		,		400	_	-
2024 - 06 - 01		Product_H	P0002	2024 - 08 - 13		
	400			2024 - 06 - 20	Product_D	P0016
2024 - 08 -	-17			400		
2024 - 06 - 02		Product_H	P0002	2024 - 08 - 13		



2024-06-21	Product_D	P0016	50 2024-07-29		
2024 - 08 - 13 $2024 - 06 - 22$	Product D	P0016		Product_C	P0005
400 2024-08-13			2024 - 07 - 29		P0007
2024-06-23	Product_D	P0016	2024-09-02		<b>D</b> 0000
2024 - 08 - 13 $2024 - 06 - 24$ $400$	Product_D	P0016	2024-06-12 50 2024-07-17		P0009
2024-08-13 2024-06-25	Product_D	P0016		Product_I	P0009
$ \begin{array}{r} 200 \\ 2024 - 08 - 13 \end{array} $				Product_I	P0009
2024-06-26 400 2024-09-09	Product_C	P0027	50 2024-07-17 2024-06-15		P0009
2024-09-09 2024-06-27 400	Product_C	P0027	50 2024-07-17	0	F0009
2024 - 09 - 09 $2024 - 06 - 28$	Product_C	P0027	2024-06-16	Product_I	P0009
400 2024-09-09		D0007	2024-07-17 2024-06-17	Product_I	P0009
2024-06-29 400 2024-09-09	Product_C	P0027	$ \begin{array}{r} 10 \\ 2024 - 07 - 17 \\ 2024 - 06 - 18 \end{array} $		P0013
Plant_Z	Product Name	Order	50 2024-08-21	0	10013
DueDate	uantity (Produced)		2024-06-19	0	P0013
$\begin{array}{c} 2024 - 06 - 01 \\ 500 \\ 2024 - 07 - 10 \end{array}$	Product_A	P0003	$\begin{array}{c} 2024 - 08 - 21 \\ 2024 - 06 - 20 \\ 50 \end{array}$	Product_B	P0017
2024-07-10 2024-06-02 500	Product_A	P0003	2024-06-28 2024-06-21		P0018
$2024\!-\!07\!-\!10$	Product_A	P0003	50 2024-09-20	0	
500 2024-07-10	D 1	D0002	2024-06-22		P0018
2024-06-04 500 2024-07-10	Product_A	P0003	2024-09-20 2024-06-23 50	Product_A	P0018
2024-06-05	Product_A	P0003	2024-09-20 2024-06-24		P0018
2024-07-10 2024-06-06	Product_A	P0003	50 2024-09-20		D0026
500 2024-07-10 2024-06-07	Product_A	P0003	2024-06-25 50 2024-07-24		P0026
500 2024-07-10	Troduct_/	10003	2024-06-26	Product_E	P0026
2024-06-08	Product_A	P0003	2024-07-24 2024-06-27	Product_F	P0032
2024-07-10 2024-06-09	Product_C	P0005	$ \begin{array}{r} 10 \\ 2024 - 09 - 27 \end{array} $		



```
Product C
2024 - 06 - 28
                               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 %
```