**Додаток A. Лістинг**

**class Test(object):**

**@classmethod**

**def main(cls, args):**

**GRAPH\_MATRIX = [**

**[0, 0, 0, 0, 5, 0, 0, 0, 0, 0],**

**[0, 0, 0, 0, 4, 3, 0, 0, 0, 0],**

**[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],**

**[0, 0, 0, 0, 0, 0, 3, 0, 0, 0],**

**[0, 0, 0, 0, 0, 0, 0, 7, 0, 0],**

**[0, 0, 0, 0, 0, 0, 0, 3, 0, 0],**

**[0, 0, 0, 0, 0, 0, 0, 4, 5, 0],**

**[0, 0, 0, 0, 0, 0, 0, 0, 0, 9],**

**[0, 0, 0, 0, 0, 0, 0, 0, 0, 5],**

**[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],**

**]**

**TASKS\_WEIGHTS = [8, 5, 6, 5, 6, 4, 5, 10, 6, 6]**

**System = System(GRAPH\_MATRIX, TASKS\_WEIGHTS)**

**System.run()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**import sys**

**Test.main(sys.argv)**

**class Task(object):**

**id = int()**

**duration = int()**

**remainder = int()**

**dependentTasks = ()**

**fatherTasks = ()**

**fatherTasksToDo = ()**

**isPretended = False**

**def \_\_init\_\_(self, id, duration):**

**self.id = id**

**self.duration = duration**

**self.remainder = duration**

**def getId(self):**

**return self.id**

**def getDuration(self):**

**return self.duration**

**def getRemainder(self):**

**return self.remainder**

**def setDependentTasks(self, dependentTasks):**

**self.dependentTasks = dependentTasks**

**def getDependentTasks(self):**

**return self.dependentTasks**

**def getFatherTasksToDo(self):**

**return self.fatherTasksToDo**

**def setFatherTasks(self, fatherTasks):**

**self.fatherTasks = fatherTasks**

**self.fatherTasksToDo = fatherTasks.clone()**

**def isPretended(self):**

**return self.isPretended**

**def setPretended(self, pretended):**

**self.isPretended = pretended**

**def execute(self):**

**self.remainder -= 1**

**import csv**

**class System(object):**

**PROCESSOR\_COUNT = 3**

**marker = 0**

**Processors = []**

**tasks = Map()**

**isBusFree = True**

**Connections = ()**

**readyTasks = ()**

**executedTasks = ()**

**numberOfTasks = int()**

**doneTasks = ()**

**def \_\_init\_\_(self, graphMatrix, tasksWeights):**

**len(graphMatrix)**

**setSystemParametres(graphMatrix, tasksWeights)**

**self.Processors = [None]\*PROCESSOR\_COUNT**

**i = 0**

**while i < self.PROCESSOR\_COUNT:**

**self.Processors[i] = Processor**

**i += 1**

**def run(self):**

**self.marker = 0**

**takt = 0**

**self.marker = 0**

**self.Processors[0].setMarkered(True)**

**modelingResult = ArrayList()**

**taktLine = ""**

**split = ";"**

**taktLine += "t" + split + " "**

**i = 1**

**while i <= self.PROCESSOR\_COUNT:**

**taktLine += "P" + i + split + " "**

**i += 1**

**taktLine += " Bus"**

**print taktLine**

**modelingResult.add(taktLine)**

**while len(self.doneTasks) != self.numberOfTasks:**

**takt += 1**

**while len(Processors):**

**if (self.Processors[i].isBusy()) and (self.Processors[i].getExecuteTask().getRemainder() == 0):**

**self.Processors[i].setBusy(False)**

**self.Processors[i].addExecutedTasks()**

**self.doneTasks.add(self.Processors[i].getExecuteTask())**

**self.executedTasks.remove(self.Processors[i].getExecuteTask())**

**while j < self.numberOfTasks:**

**if (not self.doneTasks.contains(self.tasks.get(j))) and (not self.executedTasks.contains(self.tasks.get(j))):**

**if self.tasks.get(j).getFatherTasksToDo().contains(self.Processors[i].getExecuteTask()):**

**self.tasks.get(j).getFatherTasksToDo().remove(self.Processors[i].getExecuteTask())**

**if (self.tasks.get(j).getFatherTasksToDo().size() == 0) and (not self.readyTasks.contains(self.tasks.get(j))):**

**self.readyTasks.add(self.tasks.get(j))**

**j += 1**

**i += 1**

**while i < self.PROCESSOR\_COUNT:**

**if not self.Processors[i].isBusy():**

**if len(self.readyTasks) != 0:**

**while j < len(self.readyTasks) - 1:**

**while k < j:**

**if self.readyTasks.get(k).getDuration() > self.readyTasks.get(j).getDuration():**

**self.readyTasks.add(j, self.readyTasks.get(k))**

**self.readyTasks.remove(k + i)**

**k += 1**

**j += 1**

**while j < len(self.readyTasks):**

**if self.Processors[i].pretends(self.readyTasks.get(j), self.executedTasks):**

**if taskForExecute != None:**

**if self.Processors[i].getExecuteTask() != None:**

**if getConnectionWeight(self.readyTasks.get(j), self.Processors[i].getExecuteTask()) > getConnectionWeight(taskForExecute, self.Processors[i].getExecuteTask()):**

**taskForExecute = self.readyTasks.get(j)**

**else:**

**taskForExecute = self.readyTasks.get(j)**

**j += 1**

**if self.Processors[i].isMarkered() & (not self.isBusFree):**

**self.isBusFree = False**

**if taskForExecute != None:**

**self.Processors[i].setExecuteTask(taskForExecute)**

**self.executedTasks.add(taskForExecute)**

**self.readyTasks.remove(taskForExecute)**

**else:**

**self.Processors[i].setExecuteTask(self.readyTasks.get(0))**

**self.executedTasks.add(self.readyTasks.get(0))**

**self.readyTasks.remove(0)**

**self.Processors[i].setBusy(True)**

**i += 1**

**while len(Processors):**

**if self.Processors[i].isBusy():**

**self.Processors[i].getExecuteTask().execute()**

**i += 1**

**while i < self.PROCESSOR\_COUNT:**

**if (self.Processors[i].isMarkered()) & (self.isBusFree):**

**self.Processors[i].setMarkered(False)**

**if i != self.PROCESSOR\_COUNT - 1:**

**self.Processors[i + 1].setMarkered(True)**

**self.marker = i + 1**

**break**

**else:**

**self.Processors[0].setMarkered(True)**

**self.marker = 0**

**break**

**i += 1**

**taktLine = takt + split + " "**

**while i < self.PROCESSOR\_COUNT:**

**if self.Processors[i].isBusy():**

**Task = self.Processors[i].getExecuteTask()**

**if Task != None:**

**taktLine += (self.Processors[i].getExecuteTask().getId() + 1)**

**else:**

**taktLine += "-"**

**if self.Processors[i].isMarkered():**

**taktLine += "\*" + split + " "**

**else:**

**taktLine += split + " "**

**i += 1**

**if not self.isBusFree:**

**taktLine += self.Connections.get(self.marker).getFromTask().getId() + " -> " + self.Connections.get(self.marker).getToTask().getId() + split**

**print taktLine**

**modelingResult.add(taktLine)**

**writeToFile(modelingResult)**

**def setSystemParametres(self, graphMatrix, tasksWeights):**

**self.tasks = {}**

**i = 0**

**while i < self.numberOfTasks:**

**self.tasks.put(i, Task)**

**i += 1**

**i = 0**

**while i < self.numberOfTasks:**

**while j < self.numberOfTasks:**

**if graphMatrix[i][j] != 0:**

**dependentTasks.add(self.tasks.get(j))**

**j += 1**

**self.tasks.get(i).setDependentTasks(dependentTasks)**

**i += 1**

**i = 0**

**while i < self.numberOfTasks:**

**while j < self.numberOfTasks:**

**if graphMatrix[j][i] != 0:**

**fatherTasks.add(self.tasks.get(j))**

**j += 1**

**self.tasks.get(i).setFatherTasks(fatherTasks)**

**i += 1**

**self.Connections = ArrayList()**

**i = 0**

**while i < self.numberOfTasks:**

**while j < self.numberOfTasks:**

**if graphMatrix[i][j] != 0:**

**self.Connections.add(Connection)**

**j += 1**

**i += 1**

**i = 0**

**while i < self.numberOfTasks:**

**if isTopTask(graphMatrix, self.tasks.get(i)):**

**self.readyTasks.add(self.tasks.get(i))**

**i += 1**

**def isTopTask(self, graphMatrix, task):**

**i = 0**

**while len(graphMatrix):**

**if graphMatrix[i][task.getId()] != 0:**

**return False**

**i += 1**

**return True**

**def getConnectionWeight(self, task1, task2):**

**i = 0**

**while i < len(self.Connections):**

**if (self.Connections.get(i).getFromTask() == task1) and (self.Connections.get(i).getToTask() == task2):**

**return self.Connections.get(i).getWeight()**

**if (self.Connections.get(i).getFromTask() == task2) and (self.Connections.get(i).getToTask() == task1):**

**return self.Connections.get(i).getWeight()**

**i += 1**

**return -1**

**def writeToFile(self, modelingResult):**

**fileName = "result.csv"**

**try:**

**with open(fileName, 'wb') as csvfile:**

**spamwriter = csv.writer(csvfile, delimiter=' ',**

**quotechar='|', quoting=csv.QUOTE\_MINIMAL)**

**for s in modelingResult:**

**spamwriter.writerow(s + "\n")**

**except IOException, err:**

**print err**

**class Processor(object):**

**isBusy = bool()**

**executeTask = Task()**

**isMarkered = bool()**

**executedTasks = ()**

**def \_\_init\_\_(self, busy, markered):**

**self.isBusy = busy**

**self.isMarkered = markered**

**def isBusy(self):**

**return self.isBusy**

**def setBusy(self, busy):**

**self.isBusy = busy**

**def isMarkered(self):**

**return self.isMarkered**

**def setMarkered(self, markered):**

**self.isMarkered = markered**

**def getExecuteTask(self):**

**return self.executeTask**

**def setExecuteTask(self, executeTask):**

**self.executeTask = executeTask**

**def getExecutedTasks(self):**

**return self.executedTasks**

**def addExecutedTasks(self):**

**self.executedTasks.add(self.executeTask)**

**def pretends(self, Task, allExecutedTasks):**

**i = 0**

**while i < len(self.executedTasks):**

**taskList = self.executedTasks.get(i).getDependentTasks();**

**while j < len(allExecutedTasks):**

**if taskList.contains(allExecutedTasks.get(j)):**

**taskList.remove(allExecutedTasks.get(j))**

**j += 1**

**if taskList.contains(Task):**

**return True**

**i += 1**

**return False**