**Subalgorithm specification**

subalg addEmployee(name, age, degree, salary):

Data: String name, int age, String degree, float Salary

Result: An employee is created and added to the repository

Pre: name is a nonempty string, age is a valid integer number > 0, degree is a nonempty string, salary is a float number > 0

Post: An employee with the given details is added

subalg remveEmployee(employee):

Data: Employee employee

Result: The employee is removed from the repository

Pre: employee is an Employee instance that is already part of the repository

Post: The employee is removed

func getAllEmployees()

Data:

Result: The function returns an array containing Employee instances

Pre:

Post: The returned array contains the Employee instances from the repository

func filterEmployeesByAge(min, max):

Data: int min, int max

Result: The function returns an array containing Employee instances

Pre: min and max should be valid integer numbers > 0 and min < max

Post: The returned array contains the Employee instances from the repository that have their age between min and max

array filteredDatasource

for employee in repository.getAllElements

begin

if employee.age > min and employee.age < max

filteredDatasource.add(employee)

endif

endfor

filterEmployeesByAge -> filteredDatasource

endfunc

func filterEmployeesByDegree(degree):

Data: String degree

Result: The function returns an array containing Employee instances

Pre: degree is a nonempty string

Post: The returned array contains the Employee instances from the repository that have their degree equal to degree

array filteredDatasource

for employee in repository.getAllElements

begin

if employee.degree = degree

filteredDatasource.add(employee)

endif

endfor

filterEmployeesByDegree -> filteredDatasource

endfunc