**algorithmization and programming**

**final exam var 1**

ACTIVITIES

Напишите программы согласно вашему варианту

**Сохраните все коды задач в одном txt файле. Назовите файл вашим именем, фамилией и вариантом и загрузите в платонус (ex. Balgabek\_Askar\_1variant.txt)**

**Обратите внимание, после загрузки файла нужно подтвердить его отправку. Необходимо отправить на проверку, иначе ваш вариант не будет виден проверяющему преподавателю**

Grades: One task – 50 points, two tasks – 80 points, three tasks – 100 points

Также, вы можете решать задачи в стиле ооп или через структуры. В случае решения через структуры, вы теряете 5 баллов с каждой задачи. Решение в стиле ооп подразумевает что все действия для решения задач необходимо поместить во внутренние функции классов. Также, следуйте принципам инкапсуляции (геттеры и сеттеры).

**Variant1**

1. Создайте структуру/класс **Candidate**{

**name** - имя

**money** - количество денег у кандидата

**percent** - процент пожертвований с денег кандидата

**elite** – является ли кандидат элитным членом

}

Вы собираетесь открыть тайное общество и набираете участников. Введите переменную **sum** – количество денег, нужных для открытия общества. Введите количество кандидатов и затем введите массив кандидатов.

Каждый кандидат платит за взнос определённый процент (**percent**) от своих денег (**money**)

Если кандидат предлагает меньше 10 процентов от необходимой суммы(**summ**), то вы не принимаете его в клуб.

Если это кандидат является элитным членом (**elite**) то вы берете его в любом случае, и не берете с него плату

Пробегитесь по списку кандидатов и наберите подходящих. После посчитайте общую сумму взносов каждого принятого члена, и проверьте, достаточно ли этих денег чтобы открыть клуб (**summ**). Если достаточно, то выведите имена членов, а также общую сумму денег. Если нет, то выведите на экран надпись о том, что общество не открыто.

(совет1: используйте вектора. В них удобно добавлять принятых кандидатов)

**Example:**

**Input:**

10000

7

Johny 20000 10 0

Adam 10000 5 0

Sally 11000 20 1

Malkolm 5000 10 0

Ron 4000 100 0

Jessica 30000 15 0

Leon 50000 20 1

**Notes:**

Johny платит 10 процентов от 20000, это 2000. Это больше чем 1000 (10 процентов от 10000). Значит его берем и в копилку 2000. Адам платит 5 процентов от 10000, это 500, что меньше чем 1000. Его не берем. Sally элитный член. Ее берем и не добавляем в копилку ничего. По итогу в общем в копилке 10500, что достаточно чтобы открыть общество.

**Output:**

10500

Johny

Sally

Ron

Jessica

Leon

1. Создайте структуру/класс **Company**{

**name** – название компании

**company\_type** – тип проектов на которых специализируется компания

**budget** – количество денег у компании

**percent** - процент налога, которая компания платит вам

}

Создайте структуру/класс **Project**{

**name** – название проекта

**project\_type** – тип проекта

**cost** – стоимость проекта

**income** - доход с проекта

}

Вы теневой владелец нескольких подозрительных компаний. Введите количество компаний и создайте массив из компаний (**Company**). После введите количество проектов и создайте массив из проектов (**Project**). Это два отдельных вектора/массива.

Каждая компания имеет тип проектов, которые она берет на выполнение (**company\_type**), свой бюджет, который она тратит на проекты (**budget**), а также, каждая компания платит свой налог с прибыли проекта (**percent**). Каждый проект имеет свой тип (**project\_type)**, стоимость за выполнение (**cost**) и прибыль, которую получит компания, выполнив его (**income**).

Каждая компания по очереди набирает проекты из общего списка. Компания берет только те проекты, чей тип (**project\_type**) совпадает с типом самой компании (**company\_type**). На каждый проект она тратит свой бюджет (**budget**). То есть, если, проходя по списку, бюджета компании не хватит чтобы взять новый проект (**cost**), компания перестает набирать проекты.

Каждый проект приносит компании прибыль (**income**), и компания платит вам налог с выполнения проекта (**percent**).

Если компания после вычета налогов с каждого проекта суммарно заработала меньше чем потратила, вы закрываете компанию.

Выведите на экран все компании, которые вы закрыли и общую сумму налогов, которые вы собрали.

(ВАЖНО1: Если делайте структуры, то теряете 5 баллов)

(ВАЖНО: сделайте в стиле ООП чтобы получить максимум баллов. Для этого используйте классы, и все вычисления должны быть в функциях внутри классов, также следуйте принципу инкапсуляции)

(совет1: используйте вектора. В них удобно добавлять принятых кандидатов)

**Example:**

**Input:**

3

AirCorp Fly 1000 20

DresdenInc Building 4000 10

TarbiSoil Oil 2000 30

9

ProjectA Fly 500 500

ProjectB Oil 1000 1500

ProjectC Fly 300 400

ProjectDog Fly 500 4000

ProjectE Building 1500 3000

Project8 Building 2000 1000

ProjectG Building 500 1500

ProjectH Oil 1000 1000

ProjectNo Building 1500 10000

**Notes:**

AirCorp берет projectA и projectС. После этого у нее остается 200 а projectDog стоит 500 значит она не может его взять. По итогу 2 проекта приносят ей 900 прибыли, с которых она платит 20 процентов налогов = 180. После вычета налогов 900 – 180 = 720, а потратила она 800. Что значит что это убыточная компания и она закрывается. DresdenInc заработал 4950 а потратил 4000, значит компания остается. TarbiSoil заработал 1750 а потратил 2000 и компания тоже закрывается. В общем налоги со всех компаний 180 + 550 + 750 = 1480

**Output:**

1480

AirCorp

TarbiSoil

1. Создайте структуру/класс **Colony**{

**name** – название колонии

**max\_plan** – максимальное количество туннелей, запланированных колонией

**battery\_amount** – общее количество электрических батарей в колонии

**ants\_number** – количество муравьев

**ants** - массив/вектор из муравьев

}

Создайте структуру/класс **Ant**{

**name** – название муравья

**productivity** – сколько туннелей в день копает муравей

**battery\_spent** – сколько батарей он потребляет в день

**hacked** – взломан ли муравей

}

Вы запускаете на марсе N колоний КиберМуравьев для облагораживания планеты. Каждая колония должна вырыть сеть туннелей под землей.

У каждой колонии есть план по количеству туннелей (**max\_plan**) по достижению которого она больше не копает новые туннели.

Также каждая колония снабжена определенным количеством батарей(**battery\_amount**) для подпитки работников. И в каждой колонии есть бригада киберМуравьев (**ants**), которые копают туннели

Каждый муравей потребляет определенное количество батарей в день (**battery\_spent**), и копает определенное количество туннелей в день (**productivity**)

Муравьи в бригадах копают по очереди. Сначала первый потом второй и так далее

Если муравей взломан (**hacked**), то вместо того чтобы копать - он закапывает существующие ходы

Посчитайте сколько туннелей в целом получится выкопать пока во всех колониях не закончится еда или они не достигнут лимита

(ВАЖНО1: Если делайте структуры, то теряете 5 баллов)

(ВАЖНО: сделайте в стиле ООП чтобы получить максимум баллов. Для этого используйте классы, и все вычисления должны быть в функциях внутри классов, также следуйте принципу инкапсуляции)

**Example:**

**Input:**

3

ColonyA 100 30 4

AntAlpha 10 2 0

AntBeta 15 4 0

AntGamma 20 3 1

AntOmega 10 6 0

Colony31 50 40 3

Antihero 10 5 0

Antique 15 10 0

Antigua 20 5 0

ColonyTruck 200 30 5

Zalant 20 4 0

Yormungant 10 2 1

Bubblegant 15 5 0

Ultrant 12 4 0

Lablant 5 3 1

**Notes:**

ColonyA обладает 28 батареями. AntAlpha копает 10 тунелей и ест 2 батареи, потом AntBeta копает 15 и ест 4, затем AntGamma, но он взломан, пожтому он разрушает 20 туннелей и ест 3 батареи, следом AntOmega копает 10 и ест 6, затем снова AntAlpha. По итогу 28 -2 - 4 - 3 - 6 -2 - 4 – 3, на втором круге AntOmega уже не хватает батарей чтобы копать. Значит в итоге туннелей 10 + 15 - 20 + 10 + 10 + 15 – 20 = 20. ColonyA выкопала 20 туннелей и у нее закончились батареи. Colony31 могла бы выкопать 90 туннелей, но у нее максимум 50, значит после 50го туннеля они остановились. ColonyTruck выкопала 57 туннелей прежде чем закончились батареи

**Output:**

127