View->Web Layout – For the testCases to appear better

PROBLEMS

# 1.

Implement class AlcoholicDrink that has info for:

* name (max of 100 chars),
* country of origin (max of 100 chars),
* percent of alcohol (float),
* base price (float).

One company have two kinds of alcoholic drinks that should be defined in two separate classes: Beer and Wine. For beer additionally keep info for the main ingredient (two possibilities: barley/wheat), and for wines keep the year of manufacturing (integer) and grapes type (max of 20 chars).

Implement in the classes:

* needed constructors and destructors (see the usage in the main function) **(5 points)**
* operator <<for printing the alcoholic drinks in format: **(5 points)**
* name country\_of\_origin computed\_price
* operator < for comparing two alcoholic drinks from any kind by their computed price **(5 points)**
* method float computePrice() for computing the price of different kind of alcoholic drinks on the following way: **(10 points)**
  + for Beer - the price is increased for 5% of the base price if the beer is originated from "Germany" and for 10% of the base price if it is made from wheat
  + for Wine - the price is increased for 5% of the base price if the wine is originated from "Italy" and for 15% of the base price if the it is manufactured before 2005

Implement global function void lowestPrice(AlcoholicDrink \*\* а, int n) that will print the alcoholic drink with lowest price. **(10 points)**

In the class AlcoholicDrink is kept an integer which is same for all alcoholic drinks and represents the discount for these products. The initial value of this variable is 5. To work with this variable implement the following static methods: **(5 points)**

* void changeDiscount(int d) to change the discount
* void total(AlcoholicDrink\*\* ad, int n) that prints the total price of all alcoholic drinks and the total price with discount.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 6 | Milenium3D 236.55  Found  Not found | Milenium3D 236.55  Found  Not found |  |
|  | 4  Kultura  2  433  Shrek  2.3  1  1 | The price for the cinema Kultura is: 419.39 | The price for the cinema Kultura is: 419.39 |  |
|  | 7 | Cheapest movie hall: Milenium3D 28.5  3D cinema halls: 2 from total 4 | Cheapest movie hall: Milenium3D 28.5  3D cinema halls: 2 from total 4 |  |
|  | 2  Vardar  3  333.3 | Initial price for the cinema with name Vardar is: 333.3 | Initial price for the cinema with name Vardar is: 333.3 |  |
|  | 5 | Cheapest movie hall: Milenium3D 236.55  3D cinema halls: 3 from total 5 | Cheapest movie hall: Milenium3D 236.55  3D cinema halls: 3 from total 5 |  |
|  | 1  Cineplexx  1  450 | Cinema created with name: Cineplexx | Cinema created with name: Cineplexx |  |
|  | 3  Milenium3D  2  395  Shrek  2.3  1 | Cinema created with name Milenium3D and discount 0.17 | Cinema created with name Milenium3D and discount 0.17 |  |

# 2.

Develop a class Bill representing a utility expense that keeps a record for: bill number (dynamically allocated array of characters), expense (integer), month and year for which is issued as a character array in form “MM.YYYY” and status (boolean representing where the bill was payed or not). **(5 points)** For this class overload:

* operator == that can compare two bills according to the bill number **(5 points)**
* operator << for printing the bill in the form: **(5 points)**

[bill number] (MM.YYYY) - [expense]

Develop a class UserProfile that will hold: a username (array of no more than 30 characters), billsArray (array of no more than 30 objects of class Bill) and number of bills in the array. For this class overload: **(5 points)**

* operator += for adding a new bill in the array of bills. If the bill was already added (a bill with that number already exist in the array) generate an exception of the class ExistingBill (that should also be defined). **(10 points)** In the case of an exception, "The bill already exists" should be printed on the screen and the duplicate bill should not be added.
* operator << that will print out all pending bills (bills which status is: not payed). "Pending bills for [username] are:", where [username] represent the name of the user that should be printed in the first line, followed by the pending bills, one per line.

Also implement function totalDue(int month, int year) that will return the total amount payed for the given month. **(10 points)**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5  petrovska.lidija 5  254648445 2450 05.2016 1  254118445 2000 04.2016 0  2218445 1000 05.2016 0  2218445 1200 06.2016 0  212218445 1560 05.2016 0  5 2016 | -----Test all -----  The bill already exists  Unpaid bills of user petrovska.lidija are:  254118445(04.2016) - 2000  2218445(05.2016) - 1000  212218445(05.2016) - 1560  Total due for month 5 and year 2016 is:5010 | -----Test all -----  The bill already exists  Unpaid bills of user petrovska.lidija are:  254118445(04.2016) - 2000  2218445(05.2016) - 1000  212218445(05.2016) - 1560  Total due for month 5 and year 2016 is:5010 |  |
|  | 3  petrovska.lidija 4  254648445 2450 10.2016 1  254118445 2000 11.2016 0  2218445 1000 12.2016 0  212218445 1560 11.2016 0 | -----Test UserProfile & operator += &&lt;&lt; -----  Unpaid bills of user petrovska.lidija are:  254118445(11.2016) - 2000  2218445(12.2016) - 1000  212218445(11.2016) - 1560 | -----Test UserProfile & operator += &&lt;&lt; -----  Unpaid bills of user petrovska.lidija are:  254118445(11.2016) - 2000  2218445(12.2016) - 1000  212218445(11.2016) - 1560 |  |
|  | 1  254648445 2450 10.2016 1 | -----Test Bill & operator &lt;&lt;-----  254648445(10.2016) - 2450 | -----Test Bill & operator &lt;&lt;-----  254648445(10.2016) - 2450 |  |
|  | 4  petrovska.lidija 4  254648445 2450 10.2016 1  254118445 2000 11.2016 0  2218445 1000 12.2016 0  212218445 1560 11.2016 0  11 2016 | -----Test UserProfile & operator += & totalDue -----  Total due for month 11 and year 2016 is:3560 | -----Test UserProfile & operator += & totalDue -----  Total due for month 11 and year 2016 is:3560 |  |
|  | 2  254648445 2450 10.2016 1  254648445 2000 11.2016 0 | -----Test Bill & operator == -----  Equal | -----Test Bill & operator == -----  Equal |  |
|  | 2  254648445 2450 10.2016 1  254118445 2000 11.2016 0 | -----Test Bill & operator == -----  Not equal | -----Test Bill & operator == -----  Not equal |  |
|  | 4  petrovska.lidija 4  254648445 2450 05.2016 1  254118445 2000 04.2016 0  2218445 1000 04.2016 0  212218445 1560 03.2016 0  4 2016 | -----Test UserProfile & operator += & totalDue -----  Total due for month 4 and year 2016 is:3000 | -----Test UserProfile & operator += & totalDue -----  Total due for month 4 and year 2016 is:3000 |  |

Passed

# 3.

Implement a class for Character that holds single char variable with the needed constructors. (5 points)

Then implement a class CharSequence that holds dynamically allocated array of objects of class Character and the length of the array. (5 points) This class should implement needed constructors and destructor (10 points) and the following methods and operators:

* CharSequenc& operator[](int i) for accessing the i-th element of the sequence. (5 points)
* CharSequence& operator +=(const Character &c) for adding new Character object. To add the new object, expand the dynamically allocated array for +1. (10 points)
* bool operator==(const CharSequence &cs) that will compare two CharSequence objects. Two objects are equal if they have equal arrays of Character objects, and two Character objects are equal if they hold equal character. (5 points)
* int operator[](char c) that will return how many times the character c apears in the CharSequence object. (5 points)
* CharSequence toUpperCase() that will return a new CharSequence object, where all characters will be in lowercase. (10 points)

For both classes implement the operator << for printing the contents to the standard output. (5 points)

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  char\_sequence\_object | Testing CharSequence constructors  char\_sequence\_object | Testing CharSequence constructors  char\_sequence\_object |  |
|  | 4 | Testing CharSequence =operator and operator[int]  Original: Xome sequence  Copy: Some sequence | Testing CharSequence =operator and operator[int]  Original: Xome sequence  Copy: Some sequence |  |
|  | 1  x | Testing Character class  Character constructor  OK  Character operator <<  x | Testing Character class  Character constructor  OK  Character operator <<  x |  |
|  | 5  add\_these\_characters | Testing CharSequence operator+=  add\_these\_characters | Testing CharSequence operator+=  add\_these\_characters |  |
|  | 3 | Testing CharSequence copy constructor and operator[int]  Original: Xome sequence  Copy: Some sequence | Testing CharSequence copy constructor and operator[int]  Original: Xome sequence  Copy: Some sequence |  |
|  | 7 | Testing CharSequence toUpperCase  IF YOU TELL THE TRUTH, YOU DON'T HAVE TO REMEMBER ANYTHING | Testing CharSequence toUpperCase  IF YOU TELL THE TRUTH, YOU DON'T HAVE TO REMEMBER ANYTHING |  |
|  | 8 | Testing CharSequence operator ==  1  0  0 | Testing CharSequence operator ==  1  0  0 |  |
|  | 6 | Testing CharSequence operator[char]  12 | Testing CharSequence operator[char]  12 |  |

# 4.

Create a class for describing cinema. For each cinema, keep information for the following: **(5 points)**

* the name (string of up to 20 characters)
* the number of the cinema hall (integer)
* student discount (real number)
* ticket price in MKD (real numbe)

All the data in the class should be private. Student discount is the same for all objects of the class cinema and can be changed by the cinema managers. For that purpose, provide a method for changing the student discount. Student discount is expressed in percentages and the starting value is 17 percent. **(5 points)**

The basic price of one movie ticket is calculated using the method: **(5 points)**

price (bool isStudent)

- that returns the price of the ticket. If the variable isStudent is true, a student discount should also be calculated.

For the needs of CineDays, a special kind of cinema halls, 3D cinema, should be provided. For each 3D cinema, additional information is stored: **(5 points)**

* the movie name that is currently available in that hall (dynamically allocated array of characters),
* the duration of the movie in hours (real number) and
* the logical variable for whether it is a movie premiere or not (premiere - true / regular projection - false).

A method for calculating the ticket price **(5 points)** should be available for each 3D cinema:

price (bool isStudent)

- the initial price is proportionally increased depending on the duration of the movie. If the duration of the movie is longer than 2.5 hours, the initial price is increased by 50 denars. If the duration is longer than 3 hours, the initial price is increased by 70 denars. If it's a premiere, the initial price of the film is increased by 60 denars.

Also implement the following functions:

bool searchCinema (Cinema \*\* c, int n, char \* title, bool premiere, bool isStudent)

- in which the cinema halls that have the same name as the input variable will be searched. If the premiere variable is true, the search should be performed only through the 3D halls for which the projection is a premiere, otherwise all the cinema halls are searched. If the appropriate cinema is found, its name and price are printed and the method returns true. If the required cinema is not found, the method returns false. **(10 points)**

void cheapestCinema(Cinema \*\* halls, int n,bool isStudent)

- to print the name and price of the cheapest cinema in the array. Additionally, also print out how many of the cinema halls are 3D, and how many are not 3D, in the following format: **(10 points)**

[3D cinema halls: X of total Y]

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 6 | Milenium3D 236.55  Found  Not found | Milenium3D 236.55  Found  Not found |  |
|  | 4  Kultura  2  433  Shrek  2.3  1  1 | The price for the cinema Kultura is: 419.39 | The price for the cinema Kultura is: 419.39 |  |
|  | 7 | Cheapest movie hall: Milenium3D 28.5  3D cinema halls: 2 from total 4 | Cheapest movie hall: Milenium3D 28.5  3D cinema halls: 2 from total 4 |  |
|  | 2  Vardar  3  333.3 | Initial price for the cinema with name Vardar is: 333.3 | Initial price for the cinema with name Vardar is: 333.3 |  |
|  | 5 | Cheapest movie hall: Milenium3D 236.55  3D cinema halls: 3 from total 5 | Cheapest movie hall: Milenium3D 236.55  3D cinema halls: 3 from total 5 |  |
|  | 1  Cineplexx  1  450 | Cinema created with name: Cineplexx | Cinema created with name: Cineplexx |  |
|  | 3  Milenium3D  2  395  Shrek  2.3  1 | Cinema created with name Milenium3D and discount 0.17 | Cinema created with name Milenium3D and discount 0.17 |  |

Passed

# 5.

Create a structure Player with the following information:

* name (array of max 15 characters)
* current level (integer)
* points (integer).

Then create a structure ComputerGame with:

* name of the game (array of max 20 characters)
* array of players (maximum 30 players)
* number of players (integer).

Implement a function with the function signature:

void bestPlayer(computerGame \*array, int n)

This function should print the name of the player which plays the most popular game (with the greatest number of players playing it). The best player has the most points. If there are more players with the same number of points, the best one is the player which reached the highest level. The printing is in the form: “The best player XXXXXX plays the game YYYYYY”, where XXXXXX stands for the player's name, and YYYYYY stands for the game's name.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  Snake  2  ana 5 120  ivan 4 200 | Najdobar igrac e igracot so korisnicko ime ivan koj ja igra igrata Snake | Najdobar igrac e igracot so korisnicko ime ivan koj ja igra igrata Snake |  |
|  | 1  Snake  2  ana 5 120  ivan 4 120 | Najdobar igrac e igracot so korisnicko ime ana koj ja igra igrata Snake | Najdobar igrac e igracot so korisnicko ime ana koj ja igra igrata Snake |  |
|  | 2  Snake  2  user1 5 120  user2 4 120  Cars  3  koliubavi 5 130  fikjo 3 150  motor 4 140 | Najdobar igrac e igracot so korisnicko ime fikjo koj ja igra igrata Cars | Najdobar igrac e igracot so korisnicko ime fikjo koj ja igra igrata Cars |  |

# 6.

Develop a class hierarchy that can represent digital cameras. Create an abstract class Camera that represents the basics of each digital camera, holding data for:

* manufacturer (an array of no more than 20 characters)
* model (an array of no more than 20 characters)
* year of production (integer)
* resolution in mega-pixels (float)

Implement all the needed constructors, destructor and other methods needed for proper functioning **(10 points)**.

You should derive three different classes from this class: PhotoCamera, VideoCamera and FilmCamera.

For the PhotoCamera class additional information that represents where there the camera supports RAW image capturing (boolean) should be included.

For the VideoCamera class additional information representing the maximum lent of the video that can be captured (in seconds, integer) should be included.

For the FilmCamera class additional information representing the maximum frame rate (in frames per second - fps, integer) should be included.

Implement the methods price() and rentalPrice(int days) for the derived classes that return the selling and the rental price of the corresponding camera.

The prices are calculated in the following way:

* The price of PhotoCamera is calculated as: 100 + resolution\*20 (+ 50 if RAW is supported)
* The price of VideoCamera is calculated as: resolution\*80. If the camera supports capturing of videos longer than 60 minutes, the price is increased by 40%.
* The price of FilmCamera is calculated as: 50000 start price, plus if the camera supports capturing videos with frame rates faster than 30fps, than 5000 is added to the price for each fps rate above 30 that the camera supports.

The rental prices are calculated in the following way:

* For the PhotoCamera and the VideoCamera it is calculated in the same way as: 1/100 of the selling price of the camera per day. If the camera is rented for more than 7 days, a discount of 20% is calculated.
* The price for renting a FilmCamera is: 500 per day (fixed), but if the camera supports capturing with frame rates greater than 60fps, than the renting price is doubled.

Overload the operator < that will support comparing any two cameras according to their selling price.

Define a global function production(…) that will take an array of Camera pointers, a number of cameras in the array and number of days filming will last as arguments, and will return the total price for renting all the cameras in the array for the given number of days.

Also define a global function mostExpensiveCamera(…) that will take an array of Camera pointers, a number of cameras in the array as arguments, and will return the model of the camera with the highest selling price.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 4  1  Nikon  NN  2007  3  0  2  Panasonik  K444  2000  0.5  1800  3  BMD  RED  2015  80  60  1  Canon  C1  2015  5  1  5 | Price of production is: 2522.5  Most expensive camera used in production is: RED | Price of production is: 2522.5  Most expensive camera used in production is: RED |  |
|  | 1  1  Canon  D500  2011  10  1  1 | Price of production is: 3.5  Most expensive camera used in production is: D500 | Price of production is: 3.5  Most expensive camera used in production is: D500 |  |
|  | 1  2  Sony  SL400  2016  10  3555  1 | Price of production is: 8  Most expensive camera used in production is: SL400 | Price of production is: 8  Most expensive camera used in production is: SL400 |  |
|  | 1  1  Canon  D400  2011  10  0  1 | Price of production is: 3  Most expensive camera used in production is: D400 | Price of production is: 3  Most expensive camera used in production is: D400 |  |
|  | 2  1  Nikon  NN  2007  12  0  2  Panasonik  K444  2000  3  1800  80 | Price of production is: 371.2  Most expensive camera used in production is: NN | Price of production is: 371.2  Most expensive camera used in production is: NN |  |
|  | 1  3  BlackMagicDesign  Red  3  2017  100  144 | Price of production is: 144000  Most expensive camera used in production is: Red | Price of production is: 144000  Most expensive camera used in production is: Red |  |

Passed

# 7.

Define a classDonationAction that stores information for:

* name (dynamically allocated array of chars),
* id number (char array),
* collected fund value (integer),
* needed fund value (integer)

For this class implement **(15 points)**:

* constructors, destructor and the needed get-functions (see the main function for calls)
* operator<< for printing the data in the following format:

idnumber name **X** *more denars are needed*, if the needed fund is not collected, where X=needed*fund*value – collected*fund*value

idnumber name *collected*, if the needed fund is collected

* operator+= - to add a given integer to the collected fund value
* operator < - to compare two donation actions by the needed fund value that is not collected (the difference between the needed and collected fund value). If two donation actions are needed the same fund value, the actions are compared by their id number.

The id number must be string with at most 9 digits. In this class an exception should be thrown if there is an error in the id number. Two types of errors can be found: the id number contains characters that are not digits or the id number could have more than 9 digits. In the main function write a code segment that will handle with these exceptions and write an appropriate message when an error occurs : The id number has a non-digit character or The id number has more than 9 characters, accordingly. If one id number has both types of error write the second message. **(10 points)**

Define a class DonationSociety that stores information for:

* name of the president of the society (char array, max 30)
* dynamically allocated array of objects of class DonationAction
* number of donation actions (integer)

For this class implement **(25 points)**:

* constructs and desctructor (see the calls in the main function)
* functionbool **takeDonation** (char id[], int donatedFund) that takes donatedFund for the donation action with an id number id. The function will return true if the donation is taken. The donation will be taken if the assigned id number exists and in this case the value for the collected fund of the action with the given id will be updated.
* operator+= to add a new donation action in the array of donation actions, but only if there does not already exists a donation actions with the same id number. In all other cases the donation actions is not added.
* operator<< for printing the all donation actions sorted in decreasing order according to the comparison defined with the operator <. In the end, in the last row print:

President: name

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  donate lunch  12345678  450  child smile  123  30000  gift a card  aa11111111111111  599  2  1234 499  123 40000 | The id number has more than 9 characters  The donation is not taken. Wrong ID.  ===============  12345678 donate lunch 450 more denars are needed  123 child smile collected  President:Velko Velkovski | The id number has more than 9 characters  The donation is not taken. Wrong ID.  ===============  12345678 donate lunch 450 more denars are needed  123 child smile collected  President:Velko Velkovski |  |
|  | 3  donate lunch  12345678  450  child smile  123  30000  gift a card  9999  500  4  12345678 300  9999 100  9999 100  123 20000 | ===============  123 child smile 10000 more denars are needed  9999 gift a card 300 more denars are needed  12345678 donate lunch 150 more denars are needed  President:Velko Velkovski | ===============  123 child smile 10000 more denars are needed  9999 gift a card 300 more denars are needed  12345678 donate lunch 150 more denars are needed  President:Velko Velkovski |  |
|  | 2  donate lunch  12345678  450  child smile  123  30000  3  12345678 300  123 20000  123 15000 | ===============  12345678 donate lunch 150 more denars are needed  123 child smile collected  President:Velko Velkovski | ===============  12345678 donate lunch 150 more denars are needed  123 child smile collected  President:Velko Velkovski |  |
|  | 3  donate lunch  12345678  450  child smile  123  30000  gift a card  9999  500  4  12345678 300  9999 250  9999 100  123 20000 | ===============  123 child smile 10000 more denars are needed  9999 gift a card 150 more denars are needed  12345678 donate lunch 150 more denars are needed  President:Velko Velkovski | ===============  123 child smile 10000 more denars are needed  9999 gift a card 150 more denars are needed  12345678 donate lunch 150 more denars are needed  President:Velko Velkovski |  |
|  | 3  donate lunch  12345678  450  child smile  123  30000  gift a card  aa111  599  2  1234 499  123 40000 | The id number has a non-digit character  The donation is not taken. Wrong ID.  ===============  12345678 donate lunch 450 more denars are needed  123 child smile collected  President:Velko Velkovski | The id number has a non-digit character  The donation is not taken. Wrong ID.  ===============  12345678 donate lunch 450 more denars are needed  123 child smile collected  President:Velko Velkovski |  |
|  | 1  donate lunch  123456789  450  2  123456789  300  1234567890  200 | The donation is not taken. Wrong ID.  ===============  123456789 donate lunch 150 more denars are needed  President:Velko Velkovski | The donation is not taken. Wrong ID.  ===============  123456789 donate lunch 150 more denars are needed  President:Velko Velkovski |  |

Passed

# 8.

Define a structure for Driver with fields for his name (char array of max 100) and time in seconds (integer). (5 points) Then define a structure Race with fields for location (char array of max 100), number of drivers (integers) and an array of drivers (max 100). (5 points)

Write a main function that will read the data for N races and for each race will print the top 3 drivers by their time in increasing order. (30 points)

For the format of the input data and format of the output, see the sample input and sample output (the time is printed in two fixed places for minutes and two fixed places with leading zero for seconds, separated with :).

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 6  Location-1  20  Sarah  966  Natalie  1204  Nicholas  422  Nathan  294  Elizabeth  1291  Chloe  1299  James  939  Joshua  492  Andrew  1595  Alexis  1802  Matthew  975  Tyler  1551  John  1207  Ethan  1925  Emma  1246  Jacob  1822  Ashley  1344  Jonathan  878  Grace  989  Sophia  304  Location-2  14  Joshua  1365  Sarah  1701  Madison  512  Isabella  1055  Nathan  1305  Alexis  231  David  839  Brianna  511  Matthew  327  Tyler  1095  William  1469  Jacob  1547  John  1263  Emily  398  Location-3  13  Madison  1954  John  1197  Jonathan  794  Ryan  1307  David  1532  Brianna  738  Natalie  677  Joseph  1641  Alexis  642  Jacob  1410  Nicholas  1391  Ava  213  Emily  1102  Location-4  14  Joshua  1099  Sarah  380  Natalie  925  Nathan  1214  Samantha  330  Abigail  217  Matthew  769  Olivia  1305  Jacob  1243  Emma  1304  Nicholas  1446  Ava  890  Sophia  332  Emily  1634  Location-5  8  Emma  818  Andrew  1124  Alexis  729  Brianna  1826  Daniel  992  Christopher  1378  Abigail  1851  Jacob  1523  Location-6  20  Isabella  610  Chloe  539  Hannah  1946  Samantha  931  Michael  1161  Andrew  184  Alexis  1809  David  770  Daniel  1671  Matthew  1499  Joseph  380  Christopher  1140  Ryan  1630  James  428  Jacob  865  Olivia  1104  John  911  Grace  968  Ava  768  Alexander  834 | Location-1 (20 Drivers)  1. Nathan 4:54  2. Sophia 5:04  3. Nicholas 7:02  Location-2 (14 Drivers)  1. Alexis 3:51  2. Matthew 5:27  3. Emily 6:38  Location-3 (13 Drivers)  1. Ava 3:33  2. Alexis 10:42  3. Natalie 11:17  Location-4 (14 Drivers)  1. Abigail 3:37  2. Samantha 5:30  3. Sophia 5:32  Location-5 (8 Drivers)  1. Alexis 12:09  2. Emma 13:38  3. Daniel 16:32  Location-6 (20 Drivers)  1. Andrew 3:04  2. Joseph 6:20  3. James 7:08 | Location-1 (20 Drivers)  1. Nathan 4:54  2. Sophia 5:04  3. Nicholas 7:02  Location-2 (14 Drivers)  1. Alexis 3:51  2. Matthew 5:27  3. Emily 6:38  Location-3 (13 Drivers)  1. Ava 3:33  2. Alexis 10:42  3. Natalie 11:17  Location-4 (14 Drivers)  1. Abigail 3:37  2. Samantha 5:30  3. Sophia 5:32  Location-5 (8 Drivers)  1. Alexis 12:09  2. Emma 13:38  3. Daniel 16:32  Location-6 (20 Drivers)  1. Andrew 3:04  2. Joseph 6:20  3. James 7:08 |  |
|  | 4  Location-1  9  Nicholas  1156  Natalie  454  Elizabeth  1255  Michael  1956  Brianna  1838  Joseph  1090  Olivia  1908  John  1993  Alexander  996  Location-2  19  William  1647  Joseph  1084  Madison  203  Nicholas  264  Ashley  1055  Chloe  1524  Isabella  2000  James  641  Joshua  417  Abigail  1987  Grace  947  Anthony  107  Matthew  1223  Tyler  1897  Daniel  896  Jacob  1374  John  1953  Ava  1856  Sophia  1894  Location-3  7  James  725  Andrew  898  Alexis  1965  Anthony  1903  Matthew  1622  Christopher  1973  Ethan  1831  Location-4  12  Sarah  547  Madison  1480  John  982  Nathan  1375  Michael  349  David  1402  Anthony  282  Emma  881  Jonathan  815  Alyssa  1934  Sophia  754  Alexander  1068 | Location-1 (9 Drivers)  1. Natalie 7:34  2. Alexander 16:36  3. Joseph 18:10  Location-2 (19 Drivers)  1. Anthony 1:47  2. Madison 3:23  3. Nicholas 4:24  Location-3 (7 Drivers)  1. James 12:05  2. Andrew 14:58  3. Matthew 27:02  Location-4 (12 Drivers)  1. Anthony 4:42  2. Michael 5:49  3. Sarah 9:07 | Location-1 (9 Drivers)  1. Natalie 7:34  2. Alexander 16:36  3. Joseph 18:10  Location-2 (19 Drivers)  1. Anthony 1:47  2. Madison 3:23  3. Nicholas 4:24  Location-3 (7 Drivers)  1. James 12:05  2. Andrew 14:58  3. Matthew 27:02  Location-4 (12 Drivers)  1. Anthony 4:42  2. Michael 5:49  3. Sarah 9:07 |  |
|  | 5  Location-1  14  Sarah  867  Joshua  1958  Madison  1862  Isabella  588  Samantha  1454  David  1025  Andrew  1035  Abigail  971  Alyssa  1934  Brianna  1106  Ethan  902  John  1177  Grace  1283  Sophia  1295  Location-2  13  Joshua  1504  Natalie  640  Emma  813  Hannah  387  James  1645  Andrew  1323  Alexis  306  Matthew  1563  Tyler  1496  Ethan  1020  Nathan  1588  John  1627  Emily  1219  Location-3  5  Joshua  378  John  1958  Ashley  524  Grace  311  Ryan  268  Location-4  11  Alexander  123  Ashley  622  Elizabeth  619  Samantha  1121  Andrew  918  Daniel  323  Jacob  1720  Emma  1212  Nicholas  1962  Grace  1954  Emily  1376  Location-5  10  Isabella  867  Nathan  1744  Chloe  991  Mia  1518  James  337  Ryan  1004  Christopher  535  Jonathan  489  Alyssa  490  Alexander  446 | Location-1 (14 Drivers)  1. Isabella 9:48  2. Sarah 14:27  3. Ethan 15:02  Location-2 (13 Drivers)  1. Alexis 5:06  2. Hannah 6:27  3. Natalie 10:40  Location-3 (5 Drivers)  1. Ryan 4:28  2. Grace 5:11  3. Joshua 6:18  Location-4 (11 Drivers)  1. Alexander 2:03  2. Daniel 5:23  3. Elizabeth 10:19  Location-5 (10 Drivers)  1. James 5:37  2. Alexander 7:26  3. Jonathan 8:09 | Location-1 (14 Drivers)  1. Isabella 9:48  2. Sarah 14:27  3. Ethan 15:02  Location-2 (13 Drivers)  1. Alexis 5:06  2. Hannah 6:27  3. Natalie 10:40  Location-3 (5 Drivers)  1. Ryan 4:28  2. Grace 5:11  3. Joshua 6:18  Location-4 (11 Drivers)  1. Alexander 2:03  2. Daniel 5:23  3. Elizabeth 10:19  Location-5 (10 Drivers)  1. James 5:37  2. Alexander 7:26  3. Jonathan 8:09 |  |

# 9.

Create a structure (in C programming language) Film for representing films (movies). The structure should contain: **(5 points)**

* Movie title (array of 100 characters)
* Duration (integer)
* Price (decimal number)
* Is there a student discount (1-yes, 0-no)

Create a structure FilmFestival, that will contain: **(10 points)**

* Name of the festival (array of 100 characters)
* Place (where the festival is held) (array of 100 characters)
* List of films that will be presented on the festival (array of no more than 100 elements of type Film)
* Number of elements in the list (integer)

Write a function print that will print the information for each film in a given film festival **(5 points)** in the following format:

[Name of the festival] [Place]

[Title1] [Duration1] [Price1]

[Title2] [Duration2] [Price2]

...

Write a function BestFestival that accepts an array of FilmFestival variables and number of festivals in the array and will print the name of the festival that is cheapest for students and the place it is held. For the cheapest festival is considered the one that has most films that have student discount. If there is more that one festival with these characteristics than print the festival that has the longest duration of all the films. **(20 points)**

Complete functionality of the whole program **(5 points)**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  SkopjeFilmFestival  Skopje  3  Nokturama  130  200  1  Surovo  98  150  0  Vodolija  140  190  1  Cinedays  Skopje  3  Trampolina  82  150  1  Zora  85  150  1  Zaba  78  150  0 | SkopjeFilmFestival Skopje  Nokturama 130 200.00  Surovo 98 150.00  Vodolija 140 190.00  Cinedays Skopje  Trampolina 82 150.00  Zora 85 150.00  Zaba 78 150.00  Najdobar festival: SkopjeFilmFestival Skopje | SkopjeFilmFestival Skopje  Nokturama 130 200.00  Surovo 98 150.00  Vodolija 140 190.00  Cinedays Skopje  Trampolina 82 150.00  Zora 85 150.00  Zaba 78 150.00  Najdobar festival: SkopjeFilmFestival Skopje |  |
|  | 1  SkopjeFilmFestival  Skopje  2  Nokturama  130  200  1  Surovo  98  150  0 | SkopjeFilmFestival Skopje  Nokturama 130 200.00  Surovo 98 150.00  Najdobar festival: SkopjeFilmFestival Skopje | SkopjeFilmFestival Skopje  Nokturama 130 200.00  Surovo 98 150.00  Najdobar festival: SkopjeFilmFestival Skopje |  |
|  | 2  SkopjeFilmFestival  Skopje  2  Nokturama  130  200  1  Surovo  98  150  0  Cinedays  Skopje  3  Trampolina  82  150  1  Zora  85  150  1  Zaba  78  150  0 | SkopjeFilmFestival Skopje  Nokturama 130 200.00  Surovo 98 150.00  Cinedays Skopje  Trampolina 82 150.00  Zora 85 150.00  Zaba 78 150.00  Najdobar festival: Cinedays Skopje | SkopjeFilmFestival Skopje  Nokturama 130 200.00  Surovo 98 150.00  Cinedays Skopje  Trampolina 82 150.00  Zora 85 150.00  Zaba 78 150.00  Najdobar festival: Cinedays Skopje |  |

# 10.

Define a class Goal that keeps information for:

* the goal scorer (dynamically allocated array of characters),
* time (the minute) of the goal scoring event (integer),
* name of the team that scored the goal (char array max 50).

For this class implement:

* constructor with all three arguments
* operator << for printing on ostream an object of class Goal (print the minute and the name)
* operator ++ (postfix) notation for incrementing the minute for one,
* operator -- (prefix) notation for decrementing the minute for one.

Also create a class Game that keeps dynamically allocated array of objects from the class Goal and the names of the teams that play that game (two char arrays of 50). For this class implement:

* constructor with two arguments the names of the teams that play the game
* the unary operator += for adding an object of class Goal in the array of objects. If the name of the object from the class Goal is not equal to the one of the names of the teams that play the game then throw an exception of type InvalidTeamName. Handling the exception should be in the appropriate place in the main function, by printing the message Invalid team name: [the\_name]
* operator << for printing on the ostream the teams that play the game and all the scorers.

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | ARS  MUN  5  Rooney  17  MUN  Ibrahimovic  20  MUN  Ozil  44  ARS  Sanchez  77  ARS  Pogba  91  MUN | ARS - MUN  17 Rooney  20 Ibrahimovic  44 Ozil  77 Sanchez  91 Pogba | ARS - MUN  17 Rooney  20 Ibrahimovic  44 Ozil  77 Sanchez  91 Pogba |  |
|  | Barcelona  RealM  4  Messi  25  Barcelona  Ronaldo  32  RealM  Griezmann  41  Atletico  Iniesta  87  Barcelona | Invalid team name: Atletico  Barcelona - RealM  25 Messi  32 Ronaldo  87 Iniesta | Invalid team name: Atletico  Barcelona - RealM  25 Messi  32 Ronaldo  87 Iniesta |  |
|  | LIV  CHE  4  Coutinho  18  LIV  Pogba  22  MUN  Azar  29  CHE  Costa  56  CHE | Invalid team name: MUN  LIV - CHE  18 Coutinho  29 Azar  56 Costa | Invalid team name: MUN  LIV - CHE  18 Coutinho  29 Azar  56 Costa |  |

Passed

# 11.

Define a class IceCream that stores information for:

* name (dynamically allocated array of chars),
* ingredients (char array, max 100)
* price (decimal number),
* discount (integer in percentage) initial value 0.

For this class implement:

* constructors and destructor (see the main function for calls) **(5 points)**
* operator << for printing the data in the following format:

name: ingredients price [discounted price]

the part "discounted price" is printed **only if** the icecream has discount (discount value is bigger then 0) **(5 points)**

* operator ++ in prefix notation for increasing the discount for 5 **(5 points)**
* operator + for addition of object of class IceCream with char array. The addition to be implemented so the name of the new icecream is formed by concatenating the name of the icecream and the char array, separated with +. The price of the new icecream is increased for 10, and the discount and the ingredients stay the same. **(10 points)**
* method void setDiscount(int discount) that can change the value of the discount. If the value is negative or larger then 100, an **exception** should be thrown that will carry the invalid value. In the marked part of the main function write a code segment that will handle with this exception and write message Invalid discount value: [the value]. **(10 points)**
* method void setName(char\* name) that changes the name of the icecream. **(5 points)**

Create class IceCreamShop that have:

* name of the store (char array, max 50)
* dynamically allocated array of objects of class IceCream
* number of icecream objects (integer).

For this class implement:

* constructs and desctructor (see the calls in the main function)
* operator += for adding new object of the class IceCream in the field **(10 points)**
* operator << for printing a list of all icecreams in the store. First the name of the store is printed, and then the data for all icecreams in new lines. **(5 points)**

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  Caramel Apple Delight  Caramel Delight ice cream, Apples  60  Caramel Apple Delight Ice Cream in Apples  Special Ice Cream  -40 | ====== TESTING IceCream CONSTRUCTORS ======  CONSTRUCTOR  Caramel Apple Delight: Caramel Delight ice cream, Apples 60  COPY CONSTRUCTOR  Caramel Apple Delight: Caramel Delight ice cream, Apples 60  Caramel Apple Delight Ice Cream in Apples: Caramel Delight ice cream, Apples 60  OPERATOR =  Caramel Apple Delight Ice Cream in Apples: Caramel Delight ice cream, Apples 60  Special Ice Cream: Caramel Delight ice cream, Apples 60  TESTING EXCEPTION  Invalid discount value: -40 | ====== TESTING IceCream CONSTRUCTORS ======  CONSTRUCTOR  Caramel Apple Delight: Caramel Delight ice cream, Apples 60  COPY CONSTRUCTOR  Caramel Apple Delight: Caramel Delight ice cream, Apples 60  Caramel Apple Delight Ice Cream in Apples: Caramel Delight ice cream, Apples 60  OPERATOR =  Caramel Apple Delight Ice Cream in Apples: Caramel Delight ice cream, Apples 60  Special Ice Cream: Caramel Delight ice cream, Apples 60  TESTING EXCEPTION  Invalid discount value: -40 |  |
|  | 3  FINKI Ice Cream  3  Strawberry Ice Cream in Lemon Bowls  Strawberry ice cream, Lemons  100  Caramel Apple Delight  Caramel Delight ice cream, Apples  120  Dreyer’s Ice Cream Blueberry Pie  French Vanilla ice cream, Graham cracker pie crust, blueberries  60 | ====== TESTING IceCreamShop ======  CONSTRUCTOR  OPERATOR +=  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 60 | ====== TESTING IceCreamShop ======  CONSTRUCTOR  OPERATOR +=  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 60 |  |
|  | 1  Strawberry Ice Cream in Lemon Bowls  Strawberry ice cream, Lemons  80  10  Caramel Apple Delight  Caramel Delight ice cream, Apples  60  20 | ====== TESTING IceCream CLASS ======  CONSTRUCTOR  OPERATOR &lt;&lt;  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 80 (72)  Caramel Apple Delight: Caramel Delight ice cream, Apples 60 (48)  OPERATOR ++  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 80 (68)  OPERATOR +  Caramel Apple Delight + chocolate: Caramel Delight ice cream, Apples 70 (56) | ====== TESTING IceCream CLASS ======  CONSTRUCTOR  OPERATOR &lt;&lt;  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 80 (72)  Caramel Apple Delight: Caramel Delight ice cream, Apples 60 (48)  OPERATOR ++  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 80 (68)  OPERATOR +  Caramel Apple Delight + chocolate: Caramel Delight ice cream, Apples 70 (56) |  |
|  | 4  FINKI Ice Cream  3  Strawberry Ice Cream in Lemon Bowls  Strawberry ice cream, Lemons  100  Caramel Apple Delight  Caramel Delight ice cream, Apples  120  Dreyer’s Ice Cream Blueberry Pie  French Vanilla ice cream, Graham cracker pie crust, blueberries  80 | ====== TESTING IceCreamShop CONSTRUCTORS ======  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 80  FINKI fruits: strawberry ice cream, raspberry ice cream, blueberry ice cream 60  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 80 | ====== TESTING IceCreamShop CONSTRUCTORS ======  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 80  FINKI fruits: strawberry ice cream, raspberry ice cream, blueberry ice cream 60  FINKI Ice Cream  Strawberry Ice Cream in Lemon Bowls: Strawberry ice cream, Lemons 100  Caramel Apple Delight: Caramel Delight ice cream, Apples 120  Dreyer’s Ice Cream Blueberry Pie: French Vanilla ice cream, Graham cracker pie crust, blueberries 80 |  |

Passed

# 12.

Investment fund company has a need of a new program that will automate and help the process. For this purpose you need to implement the following classes. The class InvestmentPlan with the following data:

* name of the investor (char array max 80)
* the plan or the sum of investment (real number)
* shares number (real number)
* bool variable representing if the plan is from the current day. (true - current day payment, false - old payment)

Additionally, you need to implement abstract class InvestmentFund with the following data members:

* name of the fund (char array max 50)
* dynamically allocated array of investment plans (objects of the classInvestmentPlan).
* number of investment plans (integer representing the size of the investment plans array)
* price of share (real number) **(5 points)**

For classes InvestmentFund and InvestmentPlan define the needed constructors, destructors and и get() or set() methods. For the class InvestmentFund you need to implement pure virtual function double profit() that will compute the daily profit of the investment fund. **(10 points)**.

The fund manages more funds that can belong in one of two types: money fund and shares fund. Design the classes MoneyFund and ActionFund **(5 points)**

For the money fund keep one static variable:

* yearly provision in percents (real number in the range 0-100). The initial value is 1%, and can not be changed.

For the action fund keep two static variables:

* yearly provision in percents (real number in the range 0-100). The initial value is 1.5%, and can not be changed.
* entry provision in percents (real number in the range 0-100). The initial value is 3%, and can not be changed.

The daily profit for the money funds is computed by the formula:

(total sum of investments \* provision / 100) / 365 **(10 points)**

Daily profit from the action funds is the sum of the profit from the yearly provision and the entry provision computed by the following formula:

(total sum of investments \* provision / 100) / 365 + value of new plans \* entry provision / 100` **(15 points)**

The total sum of investments of a fund is computed as a sum of all investments in that fund multiplied by the price of the share. In the formula in the action funds, the value of the new plans is computed as a sum of all investments where the value of the bool variable is true.

Implement separate function double totalProfit(InvestmentFund\*\* funds, int n) that as arguments accepts array of pointers to investment funds array and size of the array, and as a result should return the total daily profit of the fund. It is computed as a sum of daily profits of all funds. **(5 points)**.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  KBPublikum  4  Aleksandar  1500000  3000  Marko  1500000  2000  Atanas  2000000  4000  Suzana  2000000  3000  110  KBInvest  3  Tankica  500000  500  Elena  600000  1000  Natasha  300000  800  120 | 18047.5 | 18047.5 |  |
|  | 2  WVPInvest  5  Aleksandar  1500000  4000  Marko  1500000  3000  Atanas  2000000  6000  Suzana  2000000  5000  Maja  1000000  2000  110  WVPCash  3  Tankica  500000  1000  Elena  600000  1100  Natasha  300000  1000  120 | 18075.6 | 18075.6 |  |
|  | 4  IlirikaBRIK  2  Adrijan  500000  150  Arton  550000  250  100  IlirikaBlueChip  5  Dario  300000  100  Gjorgji  300000  150  Tomi  400000  200  Monika  350000  100  Bojana  250000  100  101  IlirikaDAX  3  Angel  1000000  1000  Mirce  500000  1000  Emilija  700000  1100  200  IlirikaSP500  3  Spase  500000  1500  Zorica  600000  1600  Maja  600000  1700  155 | 37551.4 | 37551.4 |  |

Passed

# 13.

Define class Pasta that has info for:

* name (dynamically allocated array of characters),
* price (float),
* discount (integer in percents) initial value 0.
* calories (integer)

In the class implement:

* needed constructors and destructors (see the usage in the main function) **(5 points)**
* method print() for printing the data on the OS in the following format:
* name calories price (price on discount)

– part in the price on discount is printed only if the pasta has discount (the value of discount is larger than zero) **(5 points)**

* method dby5() for decreasing the discount for 5% **(5 points)**
* method light() - from pasta is subtracted integer of calories, the name of the new pasta wet by concatenating Light to the old name, calories are subtracted for the passed value, and the price is decreased by 20%. **(10 points)**
* method void setDiscount(int discount) that can change the value of the discount. **(10 points)**
* method void setName(char\* name) that changes the name of the pasta. **(5 points)**

Create class ItalianRestaurant that has:

* name (max of 50 chars)
* dynamically allocated array of objects from class Pasta
* number of pasta (integer).

Implement in the class:

* needed constructors and destructors (see the usage in the main function)
* method add(Pasta& p) for adding new object from class Pasta in the array **(10 points)**
* method print() for printing a list of all pasta in the restaurant. First is printed the name of the restaurant, and then all pasta in new lines. **(5 points)**
* method void setPasta(int i, Pasta &p) that changes the pasta at index i in the array with pasta object p **(5 points)**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  Gino  3  Bolognese  200  493  Carbonara  220  307  QuatroFormagi  300  375 | ====== TESTING Italian Restaurant ======  CONSTRUCTOR  Add  Gino  Bolognese 493 200  Carbonara 307 220  QuatroFormagi 375 300 | ====== TESTING Italian Restaurant ======  CONSTRUCTOR  Add  Gino  Bolognese 493 200  Carbonara 307 220  QuatroFormagi 375 300 |  |
|  | 2  Siciliana  280  10  427  QuatroFormagi  Carbonara  -40 | ====== TESTING Pasta CONSTRUCTORS ======  CONSTRUCTOR  Siciliana 427 280  COPY CONSTRUCTOR  Siciliana 427 280  QuatroFormagi 427 280  OPERATOR =  QuatroFormagi 427 280  Carbonara 427 280 | ====== TESTING Pasta CONSTRUCTORS ======  CONSTRUCTOR  Siciliana 427 280  COPY CONSTRUCTOR  Siciliana 427 280  QuatroFormagi 427 280  OPERATOR =  QuatroFormagi 427 280  Carbonara 427 280 |  |
|  | 1  Bolognese  200  10  493  Carbonara  220  20  307  50 | ====== TESTING Pasta CLASS ======  CONSTRUCTOR  Print  Bolognese 493 200 (180)  Carbonara 307 220 (176)  Decrease by 5  Bolognese 493 200 (190)  Calories  BologneseLight 443 160 | ====== TESTING Pasta CLASS ======  CONSTRUCTOR  Print  Bolognese 493 200 (180)  Carbonara 307 220 (176)  Decrease by 5  Bolognese 493 200 (190)  Calories  BologneseLight 443 160 |  |
|  | 4  Gino  3  Bolognese  200  493  Carbonara  220  307  QuatroFormagi  300  375 | ====== TESTING Italian Restaurant CONSTRUCTORS ======  Gino  Funghi 275 260  Carbonara 307 220  QuatroFormagi 375 300  Gino  Bolognese 493 200  Carbonara 307 220  QuatroFormagi 375 300 | ====== TESTING Italian Restaurant CONSTRUCTORS ======  Gino  Funghi 275 260  Carbonara 307 220  QuatroFormagi 375 300  Gino  Bolognese 493 200  Carbonara 307 220  QuatroFormagi 375 300 |  |

# 14.

You are given a class Person that stores info for name and surname (char array) and year of birth (integer).

You are given a class Song that stores info for: name of the song (char array), performers of the song (array of max 10 objects of class Person), number of performers, and ratings of the song (dynamically allocated array of integers) and count of people that gave ratings. A rating is a number from 1 to 10.

You are also given a class Movie that has info for name of the movie (dynamically allocated array of characters) and popularity of the movie (integer from 1 to 10).

Create a class MovieSong that represents a song that is a soundtrack of a movie (class that is derived from the classes Song and Movie) so the objects from this class contain info for the name of the song, performers of the song, number of performers, grades from people, number of grades, name of the movie, popularity of the movie and popularity of the song (integer from 1 to 10). **(5 points)**

Each song should enable the following methods:

* float getTotalRating() - that returns the total rating of the song. For each song that is not a movie song, the rating is an average of the minimum and maximum rating given from people. The rating of a movie song is computed as a product of the average of minimum and maximum rating given from people and a coefficient of popularity (computed by dividing the popularity of the song with the popularity of the movie). In a case when a song does not have any ratings an exception from the class RatingMissingException should be thrown **(15 points)**.
  + print() - that prints the name of the song and the performers of the song, and if the song is a movie song, in new line prints the name of the movie. **(10 points)**.

The format of printing a movie song is:

[Performer 1] ([Year1]), [Performer2] ([Year2]) … : [Name]

Movie: [Name of the movie]

Implement the following global functions:

* double averageRating(Song\*\* songs, int n) that returns the average rating of all movie songs from the list of n songs. For the songs without any ratings from people, the default rating is assumed to be 5. **(15 points)**
* void printSongs(char\* performer, Song\*\* songs, int n) - that for a given array of songs, will print only those performed by the performer passed as an argument. **(10 points)**

Fully functionality of the program. **(5 points)**

**You are allowed to add additional methods to the provided classes Person, Song and Movie.**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  Glory 2 J.Legend 1978 Common 1972 2 4 10 Selma 30 5 | -----TEST MovieSong-----  Object of class MovieSong is created | -----TEST MovieSong-----  Object of class MovieSong is created |  |
|  | 5  Glory 2 J.Legend 1978 Common 1972 3 4 6 10 Selma 8 10 | -----TEST print -----  J.Legend (1978),Common (1972):Glory  Movie:Selma | -----TEST print -----  J.Legend (1978),Common (1972):Glory  Movie:Selma |  |
|  | 8  5  1 Focus 1 A.Grande 1993 5 2 7 7 10 7  2 BeautyAndTheBeast 2 J.Legend 1978 A.Grande 1993 3 8 7 10 BeautyAndTheBeast 10 8  2 Glory 2 J.Legend 1978 Common 1972 3 4 6 10 Selma 10 10  1 Heaven 1 J.Legend 1978 5 4 6 10 3 10  1 Problem 1 A.Grande 1993 5 7 7 7 7 7  J.Legend | -----TEST print songs-----  Songs of J.Legend are:  J.Legend (1978),A.Grande (1993):BeautyAndTheBeast  Movie:BeautyAndTheBeast  J.Legend (1978),Common (1972):Glory  Movie:Selma  J.Legend (1978):Heaven | -----TEST print songs-----  Songs of J.Legend are:  J.Legend (1978),A.Grande (1993):BeautyAndTheBeast  Movie:BeautyAndTheBeast  J.Legend (1978),Common (1972):Glory  Movie:Selma  J.Legend (1978):Heaven |  |
|  | 4  Glory 2 J.Legend 1978 Common 1972 3 4 6 10 Selma 8 10 | -----TEST getTotalRating-----  Total rating: 8.75 | -----TEST getTotalRating-----  Total rating: 8.75 |  |
|  | 7  4  2 BeautyAndTheBeast 2 J.Legend 1978 A.Grande 1993 3 8 7 10 BeautyAndTheBeast 10 8  2 Glory 2 J.Legend 1978 Common 1972 3 4 6 10 Selma 10 10  1 Heaven 1 J.Legend 1978 5 4 6 10 3 10  1 Problem 1 A.Grande 1993 5 7 7 7 7 7 | -----TEST average rating-----  Average rating of the songs is 6.9 | -----TEST average rating-----  Average rating of the songs is 6.9 |  |
|  | 3  Heaven 1 J.Legend 1978 5 4 6 10 3 10 | -----TEST getTotalRating-----  Total rating: 6.5 | -----TEST getTotalRating-----  Total rating: 6.5 |  |
|  | 2  Heaven 1 J.Legend 1978 5 4 6 10 3 10 | -----TEST print-----  J.Legend (1978):Heaven | -----TEST print-----  J.Legend (1978):Heaven |  |
|  | 4  Glory 2 J.Legend 1978 Common 1972 3 4 6 10 Selma 10 10 | -----TEST getTotalRating-----  Total rating: 7 | -----TEST getTotalRating-----  Total rating: 7 |  |
|  | 7  5  2 Happy 1 P.Williams 1973 0 DespicableMe2 7 10  2 BeautyAndTheBeast 2 J.Legend 1978 A.Grande 1993 3 8 7 10 BeautyAndTheBeast 10 8  2 Glory 2 J.Legend 1978 Common 1972 3 4 6 10 Selma 10 10  1 Heaven 1 J.Legend 1978 5 4 6 10 3 10  1 Problem 1 A.Grande 1993 5 7 7 7 7 7 | -----TEST average rating-----  Average rating of the songs is 6.26667 | -----TEST average rating-----  Average rating of the songs is 6.26667 |  |
|  | 6  BeautyAndTheBeast 2 J.Legend 1978 A.Grande 1993 3 8 7 10 BeautyAndTheBeast 10 8 | -----TEST Song and MovieSong-----  J.Legend (1978),A.Grande (1993):BeautyAndTheBeast  Movie:BeautyAndTheBeast  Total rating: 6.8 | -----TEST Song and MovieSong-----  J.Legend (1978),A.Grande (1993):BeautyAndTheBeast  Movie:BeautyAndTheBeast  Total rating: 6.8 |  |

Passed

# 15.

Implement class Pair that has a key (dynamically allocated array of characters) and value (integer) **(5 points)**. For this class implement:

* constructors and destructor **(5 points)**
* operator for printing on the SO << in format '[key]' -> [value] **(5 points)**
* operator -- in prefix and postfix that decreases the value for -1 **(5 points)**
* operator \* for multiplying two objects of this class and returns new object where the key is the shorter (in length) of the keys of the two objects that are multiplying, and the value is a product of the values of the two objects. If both objects has keys with same length take the right object's key in the multiply operation. **(10 points)**

Implement a class Collection where you keep dynamically allocated array of objects from the class Pair and length of the array **(5 points)**. For this class implement:

* constructors and destructor **(5 points)**
* operator += for adding new object of the class Pair in the array **(5 points)**
* operator for printing on the SO << that prints the number of elements in the array, and then each object in a new line **(5 points)**
* operator < for comparing two objects of this class using the value of the product of all pairs in the collection **(5 points)**
* method int greater(int value) that returns the count of the pairs that have value greater then the passed argument value. **(5 points)**

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | some-key 55  other-key 20  55 | ----- class Pair -----  ----- constructor -----  ----- operator << -----  'some-key' -> 55  'other-key' -> 20  ----- operator --(int) -----  'some-key' -> 55  'some-key' -> 54  ----- operator -- -----  'other-key' -> 19  ----- operator \* -----  ----- operator = -----  'some-key' -> 1026  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  'some-key' -> 54  'other-key' -> 19  'some-key' -> 1026  ----- copy constructor -----  ----- operator < -----  k >= k2  k < k2  ----- greater -----  1 | ----- class Pair -----  ----- constructor -----  ----- operator << -----  'some-key' -> 55  'other-key' -> 20  ----- operator --(int) -----  'some-key' -> 55  'some-key' -> 54  ----- operator -- -----  'other-key' -> 19  ----- operator \* -----  ----- operator = -----  'some-key' -> 1026  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  'some-key' -> 54  'other-key' -> 19  'some-key' -> 1026  ----- copy constructor -----  ----- operator < -----  k >= k2  k < k2  ----- greater -----  1 |  |
|  | key1 10  key2 15  10 | ----- class Pair -----  ----- constructor -----  ----- operator << -----  'key1' -> 10  'key2' -> 15  ----- operator --(int) -----  'key1' -> 10  'key1' -> 9  ----- operator -- -----  'key2' -> 14  ----- operator \* -----  ----- operator = -----  'key2' -> 126  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  'key1' -> 9  'key2' -> 14  'key2' -> 126  ----- copy constructor -----  ----- operator < -----  k >= k2  k < k2  ----- greater -----  2 | ----- class Pair -----  ----- constructor -----  ----- operator << -----  'key1' -> 10  'key2' -> 15  ----- operator --(int) -----  'key1' -> 10  'key1' -> 9  ----- operator -- -----  'key2' -> 14  ----- operator \* -----  ----- operator = -----  'key2' -> 126  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  'key1' -> 9  'key2' -> 14  'key2' -> 126  ----- copy constructor -----  ----- operator < -----  k >= k2  k < k2  ----- greater -----  2 |  |

Passed

# 16.

Implement class Pair that has a key (dynamically allocated array of characters) and value (real number) **(5 points)**. For this class implement:

* constructors and destructor **(5 points)**
* operator for printing on the SO << in format ('[key]', [value]) **(5 points)**
* operator ++ in prefix and postfix that increases the value for +1 **(5 points)**
* operator + for addition of two objects of this class that returns a new object in which the key is the longer (in length) of the keys of the objects that are adding and the values are added. If both objects has keys with equal length take the key from the left on in the operation addition. **(10 points)**

Implement a class Collection where you keep dynamically allocated array of objects from the class Pair and length of the array **(5 points)**. For this class implement:

* constructors and destructor **(5 points)**
* operator += for adding new object of the class Pair in the array **(5 points)**
* operator for printing on the SO << that prints the number of elements in the array, and then each object in a new line **(5 points)**
* operator == for comparing two objects of this class. Two collections are equal if they have equal number of elements and all objects on corresponding positions are equal. **(5 points)**
* method Pair\* find(const char\* key) that returns a pointer to the first objects with the provided key, if such object does not exist returns NULL pointer. **(5 points)**

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | some-key 55  other-key 20 | ----- class Pair -----  ----- constructor -----  ----- operator << -----  ('some-key', 55)  ('other-key', 20)  ----- operator ++(int) -----  ('some-key', 55)  ('some-key', 56)  ----- operator ++ -----  ('other-key', 21)  ----- operator + -----  ----- operator = -----  ('other-key', 77)  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  ('some-key', 56)  ('other-key', 21)  ('other-key', 77)  ----- copy constructor -----  ----- operator == -----  k == k2  ----- find -----  ('other-key', 21)  other-keyother not found | ----- class Pair -----  ----- constructor -----  ----- operator << -----  ('some-key', 55)  ('other-key', 20)  ----- operator ++(int) -----  ('some-key', 55)  ('some-key', 56)  ----- operator ++ -----  ('other-key', 21)  ----- operator + -----  ----- operator = -----  ('other-key', 77)  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  ('some-key', 56)  ('other-key', 21)  ('other-key', 77)  ----- copy constructor -----  ----- operator == -----  k == k2  ----- find -----  ('other-key', 21)  other-keyother not found |  |
|  | key1 10  key2 15 | ----- class Pair -----  ----- constructor -----  ----- operator << -----  ('key1', 10)  ('key2', 15)  ----- operator ++(int) -----  ('key1', 10)  ('key1', 11)  ----- operator ++ -----  ('key2', 16)  ----- operator + -----  ----- operator = -----  ('key1', 27)  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  ('key1', 11)  ('key2', 16)  ('key1', 27)  ----- copy constructor -----  ----- operator == -----  k == k2  ----- find -----  ('key2', 16)  key2other not found | ----- class Pair -----  ----- constructor -----  ----- operator << -----  ('key1', 10)  ('key2', 15)  ----- operator ++(int) -----  ('key1', 10)  ('key1', 11)  ----- operator ++ -----  ('key2', 16)  ----- operator + -----  ----- operator = -----  ('key1', 27)  ----- class Collection -----  ----- constructor -----  ----- operator += -----  ----- operator << -----  Size: 3  ('key1', 11)  ('key2', 16)  ('key1', 27)  ----- copy constructor -----  ----- operator == -----  k == k2  ----- find -----  ('key2', 16)  key2other not found |  |

Passed

# 17.

Create a program for managing products in a piano store. For that purpose, the class' Piano` *\* (5 points) \** should be created. The class should include:

* Serial number (array of 25 characters)
* Price (decimal number)
* Year of production (integer)

For this class, implement a method equals that accepts an object from the Piano class as an argument, and makes a comparison between the objects on the bases of their serial number. *\* (5 points) \**

In addition, for this class, provide a print () method for printing the object in the following format *\* (5 points) \**:

[Serial number] [Year of production] [Price]

Create a PShop class *\* (5 points) \** that contains:

* Store name (array of 50 characters)
* A list of pianos in the store (dynamically allocated array of objects of the Piano class)
* Number of objects in the array (integer)

For this class, provide the following methods:

* double value () - to calculate the current value of the pianos in the store. It is calculated as a sum of the price of all the pianos in the store *\* (5 points) \**.
* void add (Piano p) - to add a new piano in the store (adding a new object from the Piano class to the array of pianos) *\* (10 points) \**
* void sell (Piano p) to delete a given piano from the array (delete all items that are equal to the input argument, according to the serial number) *\* (10 points) \**
* void print () - to print the information about the store, so that the first row displays its name, and then in each row the information for each piano. *\* (5 points) \**

For all classes, define the necessary constructors, destructor and all other methods necessary to provide a correct program functionality. All data members in the classes are private. *\* (5 points) \**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5  FINKI-Piano  3  12345  990  2000  12346  1200  2010  12347  1300  2015  12348  950  2012  12349  1500  2017 | ---Add method---  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12348 2012 950  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12349 2017 1500 | ---Add method---  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12348 2012 950  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12349 2017 1500 |  |
|  | 3  FINKI-Piano | ---Class PShop---  FINKI-Piano | ---Class PShop---  FINKI-Piano |  |
|  | 8  FINKI-Piano  3  12345  990  2000  12346  1200  2010  12347  1300  2015  12348  950  2012  12349  1500  2017  1 | ---Full Test---  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12348 2012 950  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12349 2017 1500  Total price: 3490  FINKI-Piano  1. 12345 2000 990  2. 12347 2015 1300  Total price: 2290 | ---Full Test---  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12348 2012 950  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300  4. 12349 2017 1500  Total price: 3490  FINKI-Piano  1. 12345 2000 990  2. 12347 2015 1300  Total price: 2290 |  |
|  | 1  12345  990  2000 | ---Class Piano---  12345 2000 990 | ---Class Piano---  12345 2000 990 |  |
|  | 2  12345  990  2000 | ---Equals---  Equality  Inequality | ---Equals---  Equality  Inequality |  |
|  | 7  FINKI-Piano  3  12345  990  2000  12346  1200  2010  12347  1300  2015  1 | ---sell method---  FINKI-Piano  1. 12345 2000 990  2. 12347 2015 1300 | ---sell method---  FINKI-Piano  1. 12345 2000 990  2. 12347 2015 1300 |  |
|  | 6  FINKI-Piano  3  12345  990  2000  12346  1200  2010  12347  1300  2015 | ---value method---  Total price: 3490 | ---value method---  Total price: 3490 |  |
|  | 4  FINKI-Piano  3  12345  990  2000  12346  1200  2010  12347  1300  2015 | ---Add method---  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300 | ---Add method---  FINKI-Piano  1. 12345 2000 990  2. 12346 2010 1200  3. 12347 2015 1300 |  |

# 18.

Implement a class for online podcasts subscription (Podcast), that contains information on: **(5 поени)**

* podcast name (array of max 100 characters)
* a podcast url (a maximum of 1000 characters)
* cost for streaming (real number)
* discount with which the podcast subscription was purchased (an integer from 0 to 100).

The StreamingPodcast class is derived from the Podcast class, that will additionally store: **(5 поени)**

* monthly fee for streaming (real number).
* date when the subscription was purchased (month and year as positive integers)

Overload the print (<<) and read (>>) operators for the Podcast and StreamingPodcast classes. **(10 поени)**

Overload the operator == that will compare the podcasts of any type according to their url. **(5 поени)**

Define a Playlist class with the following information: **(5 поени)**

* the playlist name (a maximum of 100 characters)
* a collection of podcasts that are purchased by the user (dynamically allocated array of pointers to Podcast)
* total number of pointers (integer).

Overload the += operator that will enable adding a new podcast to the playlist collection. **(5 поени)** In addition, if the user has already added the same podcast, an exception of type 'ExistingPodcast' should be thrown. An appropriate constructor and method for printing a message on the screen should be provided for the ExistingPodcast class. **(5 поени)**

Create a total\_spent() method in the Playlist class that will calculate how much money is spent on the given playlist. If the podcast is purchased with a discount, the appropriate price should be calculated. If the podcast is of type 'StreamingPodcast', it is necessary to calculate the amount spent for the monthly fee (number of months times price per month). **(10 поени)**

Overload the operator << to print the playlist information, in the following format: **(5 поени)**

Playlist: Programming

Podcast: Developer Tea

URL: https://spec.fm/podcasts/developer-tea

regular price: $10

Podcast: The Game Studio

URL: https://spec.fm/podcasts/game-studio

regular price: $20, monthly fee: $1.5, purchased: 1-2018

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  The Game Studio  https://spec.fm/podcasts/game-studio  20  50 | Testing operator&gt;&gt; for Podcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, bought on discount | Testing operator&gt;&gt; for Podcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, bought on discount |  |
|  | 4  The Game Studio  https://spec.fm/podcasts/game-studio  20  0  2  6  2017 | Testing operator&gt;&gt; for StreamingPodcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $2, purchased: 6-2017 | Testing operator&gt;&gt; for StreamingPodcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $2, purchased: 6-2017 |  |
|  | 6  Programming  2  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  0  2  The Game Studio  https://spec.fm/podcasts/game-studio  20  0  1.5  1  2018 | Testing exception ExistingPodcast for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018 | Testing exception ExistingPodcast for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018 |  |
|  | 2  The Game Studio  https://spec.fm/podcasts/game-studio  20  0  1  1  2018 | Testing class StreamingPodcast and operator&lt;&lt; for StreamingPodcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1, purchased: 1-2018 | Testing class StreamingPodcast and operator&lt;&lt; for StreamingPodcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1, purchased: 1-2018 |  |
|  | 6  Programming  3  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  0  2  The Game Studio  https://spec.fm/podcasts/game-studio  20  0  1.5  1  2018  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  1 | Testing exception ExistingPodcast for Playlist  The podcast is already in the collection  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018 | Testing exception ExistingPodcast for Playlist  The podcast is already in the collection  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018 |  |
|  | 6  Programming  3  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  0  2  The Game Studio  https://spec.fm/podcasts/game-studio  20  0  1.5  1  2018  2  Developer Tea  https://spec.fm/podcasts/developer-tea  10  0  1.5  1 | Testing exception ExistingPodcast for Playlist  The podcast is already in the collection  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018 | Testing exception ExistingPodcast for Playlist  The podcast is already in the collection  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018 |  |
|  | 2  The Game Studio  https://spec.fm/podcasts/game-studio  20  17  1  6  2017 | Testing class StreamingPodcast and operator&lt;&lt; for StreamingPodcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, bought on discount, monthly fee: $1, purchased: 6-2017 | Testing class StreamingPodcast and operator&lt;&lt; for StreamingPodcast  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, bought on discount, monthly fee: $1, purchased: 6-2017 |  |
|  | 1  Bike Shed  http://bikeshed.fm/  10  0 | Testing class Podcast and operator&lt;&lt; for Podcast  Podcast: Bike Shed  URL: http://bikeshed.fm/  regular price: $10 | Testing class Podcast and operator&lt;&lt; for Podcast  Podcast: Bike Shed  URL: http://bikeshed.fm/  regular price: $10 |  |
|  | 5  Programming  1  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  0 | Testing class Playlist and operator+= for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10 | Testing class Playlist and operator+= for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10 |  |
|  | 7  Programming  2  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  0  2  The Game Studio  https://spec.fm/podcasts/game-studio  10  0  1.5  4  2018 | Testing total\_spent method() for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $10, monthly fee: $1.5, purchased: 4-2018  Total money spent: $21.5 | Testing total\_spent method() for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $10, monthly fee: $1.5, purchased: 4-2018  Total money spent: $21.5 |  |
|  | 7  Programming  2  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  50  2  The Game Studio  https://spec.fm/podcasts/game-studio  20  0  1.5  1  2018 | Testing total\_spent method() for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10, bought on discount  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018  Total money spent: $31 | Testing total\_spent method() for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10, bought on discount  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1.5, purchased: 1-2018  Total money spent: $31 |  |
|  | 1  Developer Tea  https://spec.fm/podcasts/developer-tea  20  30 | Testing class Podcast and operator&lt;&lt; for Podcast  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $20, bought on discount | Testing class Podcast and operator&lt;&lt; for Podcast  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $20, bought on discount |  |
|  | 5  Programming  2  1  Developer Tea  https://spec.fm/podcasts/developer-tea  10  0  2  The Game Studio  https://spec.fm/podcasts/game-studio  20  0  1  1  2018 | Testing class Playlist and operator+= for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1, purchased: 1-2018 | Testing class Playlist and operator+= for Playlist  Playlist: Programming  Podcast: Developer Tea  URL: https://spec.fm/podcasts/developer-tea  regular price: $10  Podcast: The Game Studio  URL: https://spec.fm/podcasts/game-studio  regular price: $20, monthly fee: $1, purchased: 1-2018 |  |

Passed

# 19.

Create a class for describing post parcels with information for:

* ID number of the parcel (integer)
* processing price (integer).

The price for processing a parcel is equal for all parcels and can be changed with the function setProcessingPrice. The initial value of this price is 150 MKD. Create a class Parcel with all required methods for full functionality of the program **(10 points)**.

A parcel can be in shape of envelope and for each envelope additional information is needed, such as the width, height of the envelope (integers) and the receiver of the envelope (char array of max 50). Create a functioning class Envelope that also implements the following method **(10 points)**:

* price() - that returns the price for sending the envelope which is calculated as: C + L \* W \* 0.3, where C is the price for processing a basic parcel while L and W are the length and the width of the envelope.

Create a class PostOffice that keeps information for at most 100 envelops and number (n<100) of envelops sent in that office. **(5 points)**

The class should implement following methods:

* int sendEnvelope(int id, char\* receiver, int length, int width) - that adds an envelop with the given arguments in the array of envelops. This method returns the price for sending the envelope. **(5 points)**
* void printEnvelopsTo(char\* receiver) - that prints the ID of the envelops that are addressed to the passed receiver. The printing is in the following format **(5 points)**:
* For [receiver] are envelops with ids: [ID1] [ID2] ...

Fully functioning program. **(5 points)**.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  4 1  5 2 V.Velkovski20b 34 6 | Parcel with id [1] is created  Price of sending envelope with ID: 2 To: V.Velkovski20b is: 211 | Parcel with id [1] is created  Price of sending envelope with ID: 2 To: V.Velkovski20b is: 211 |  |
|  | 4  1 1 V.Velkovski16a 7 21  1 2 R.BoshkovicBB 14 11  1 3 R.BoshkovicBB 11 10  2 R.BoshkovicBB | The price of sending envelope with id 1 is:194  The price of sending envelope with id 2 is:196  The price of sending envelope with id 3 is:183  For R.BoshkovicBB are envelops with ids: 2 3 | The price of sending envelope with id 1 is:194  The price of sending envelope with id 2 is:196  The price of sending envelope with id 3 is:183  For R.BoshkovicBB are envelops with ids: 2 3 |  |
|  | 5  1 5 R.BoshkovicBB 14 23  3 120  1 4 V.Velkovski16a 21 15  1 5 R.BoshkovicBB 14 23  1 6 R.BoshkovicBB 19 40 | The price of sending envelope with id 5 is:246  The price of sending envelope with id 4 is:214  The price of sending envelope with id 5 is:216  The price of sending envelope with id 6 is:348 | The price of sending envelope with id 5 is:246  The price of sending envelope with id 4 is:214  The price of sending envelope with id 5 is:216  The price of sending envelope with id 6 is:348 |  |
|  | 10  1 1 V.Velkovski16a 10 15  1 2 R.BoshkovicBB 14 13  1 3 R.BoshkovicBB 11 13  2 R.BoshkovicBB  3 120  1 4 V.Velkovski16a 10 15  1 5 R.BoshkovicBB 14 13  1 6 R.BoshkovicBB 11 13  2 R.BoshkovicBB  2 V.Velkovski16a | The price of sending envelope with id 1 is:195  The price of sending envelope with id 2 is:204  The price of sending envelope with id 3 is:192  For R.BoshkovicBB are envelops with ids: 2 3  The price of sending envelope with id 4 is:165  The price of sending envelope with id 5 is:174  The price of sending envelope with id 6 is:162  For R.BoshkovicBB are envelops with ids: 2 3 5 6  For V.Velkovski16a are envelops with ids: 1 4 | The price of sending envelope with id 1 is:195  The price of sending envelope with id 2 is:204  The price of sending envelope with id 3 is:192  For R.BoshkovicBB are envelops with ids: 2 3  The price of sending envelope with id 4 is:165  The price of sending envelope with id 5 is:174  The price of sending envelope with id 6 is:162  For R.BoshkovicBB are envelops with ids: 2 3 5 6  For V.Velkovski16a are envelops with ids: 1 4 |  |
|  | 2  1 1 V.Velkovski16a 10 15  1 2 R.BoshkovicBB 14 13 | The price of sending envelope with id 1 is:195  The price of sending envelope with id 2 is:204 | The price of sending envelope with id 1 is:195  The price of sending envelope with id 2 is:204 |  |

Passed

# 20.

Define a class Word that will have dynamically allocated array of characters. The class should only allow constructing objects by single constructor with one argument (array of characters), and if this argument is missing its value should be "UNDEFINED".

Define a class Sentence that contains:

* dynamically allocated array of objects of class Word
* number of words
* total capacity.

For this class define constructor with single argument, the max capacity of words, which will be predefined to 10 if missing. Implement the following methods:

* void add(Word w) for adding a new word in the array of words. During addition if the capacity is full, a new extended array is allocated increased for 10 new words,
* print() that will print all the words separated with one blank space,
* swap(int i, int j) that will swap the words in the array at the passed indices i and j.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 30  centigram&apos;s  removers  specious  criminologist  overachiever&apos;s  interstice  animatedly  drapery  insoles  backache  Carborundum  vizor  firming  sideswiping  tingling  silkworm  avoirdupois  chalet&apos;s  Kodachrome&apos;s  fuddled  doubloon  rector  firmly  solider  dillydallied  Schlesinger&apos;s  explosiveness  Zsigmondy  agronomists  pushy | CONSTRUCTOR  ADD WORD  PRINT SENTENCE  centigram&apos;s removers specious criminologist overachiever&apos;s interstice animatedly drapery insoles backache Carborundum vizor firming sideswiping tingling silkworm avoirdupois chalet&apos;s Kodachrome&apos;s fuddled doubloon rector firmly solider dillydallied Schlesinger&apos;s explosiveness Zsigmondy agronomists pushy  COPY  SWAP  centigram&apos;s removers specious criminologist overachiever&apos;s interstice animatedly drapery insoles backache silkworm vizor firming sideswiping tingling Carborundum avoirdupois chalet&apos;s Kodachrome&apos;s fuddled doubloon rector firmly solider dillydallied Schlesinger&apos;s explosiveness Zsigmondy agronomists pushy  ORIGINAL  centigram&apos;s removers specious criminologist overachiever&apos;s interstice animatedly drapery insoles backache Carborundum vizor firming sideswiping tingling silkworm avoirdupois chalet&apos;s Kodachrome&apos;s fuddled doubloon rector firmly solider dillydallied Schlesinger&apos;s explosiveness Zsigmondy agronomists pushy | CONSTRUCTOR  ADD WORD  PRINT SENTENCE  centigram&apos;s removers specious criminologist overachiever&apos;s interstice animatedly drapery insoles backache Carborundum vizor firming sideswiping tingling silkworm avoirdupois chalet&apos;s Kodachrome&apos;s fuddled doubloon rector firmly solider dillydallied Schlesinger&apos;s explosiveness Zsigmondy agronomists pushy  COPY  SWAP  centigram&apos;s removers specious criminologist overachiever&apos;s interstice animatedly drapery insoles backache silkworm vizor firming sideswiping tingling Carborundum avoirdupois chalet&apos;s Kodachrome&apos;s fuddled doubloon rector firmly solider dillydallied Schlesinger&apos;s explosiveness Zsigmondy agronomists pushy  ORIGINAL  centigram&apos;s removers specious criminologist overachiever&apos;s interstice animatedly drapery insoles backache Carborundum vizor firming sideswiping tingling silkworm avoirdupois chalet&apos;s Kodachrome&apos;s fuddled doubloon rector firmly solider dillydallied Schlesinger&apos;s explosiveness Zsigmondy agronomists pushy |  |
|  | 15  safflower  Penny  haven  Foreman  cutoff  stink&apos;s  Vicksburg&apos;s  photocopy  surly  felicity  exterior&apos;s  unzipping  suicides  puerility  favorite&apos;s | CONSTRUCTOR  ADD WORD  PRINT SENTENCE  safflower Penny haven Foreman cutoff stink&apos;s Vicksburg&apos;s photocopy surly felicity exterior&apos;s unzipping suicides puerility favorite&apos;s  COPY  SWAP  safflower Penny haven Foreman cutoff photocopy Vicksburg&apos;s stink&apos;s surly felicity exterior&apos;s unzipping suicides puerility favorite&apos;s  ORIGINAL  safflower Penny haven Foreman cutoff stink&apos;s Vicksburg&apos;s photocopy surly felicity exterior&apos;s unzipping suicides puerility favorite&apos;s | CONSTRUCTOR  ADD WORD  PRINT SENTENCE  safflower Penny haven Foreman cutoff stink&apos;s Vicksburg&apos;s photocopy surly felicity exterior&apos;s unzipping suicides puerility favorite&apos;s  COPY  SWAP  safflower Penny haven Foreman cutoff photocopy Vicksburg&apos;s stink&apos;s surly felicity exterior&apos;s unzipping suicides puerility favorite&apos;s  ORIGINAL  safflower Penny haven Foreman cutoff stink&apos;s Vicksburg&apos;s photocopy surly felicity exterior&apos;s unzipping suicides puerility favorite&apos;s |  |
|  | 10  repudiations  marquee&apos;s  kestrel&apos;s  harm&apos;s  Meighen&apos;s  hallucinating  Hershel  wherefore  dinginess&apos;s  Uruguay&apos;s | CONSTRUCTOR  ADD WORD  PRINT SENTENCE  repudiations marquee&apos;s kestrel&apos;s harm&apos;s Meighen&apos;s hallucinating Hershel wherefore dinginess&apos;s Uruguay&apos;s  COPY  SWAP  repudiations marquee&apos;s kestrel&apos;s hallucinating Meighen&apos;s harm&apos;s Hershel wherefore dinginess&apos;s Uruguay&apos;s  ORIGINAL  repudiations marquee&apos;s kestrel&apos;s harm&apos;s Meighen&apos;s hallucinating Hershel wherefore dinginess&apos;s Uruguay&apos;s | CONSTRUCTOR  ADD WORD  PRINT SENTENCE  repudiations marquee&apos;s kestrel&apos;s harm&apos;s Meighen&apos;s hallucinating Hershel wherefore dinginess&apos;s Uruguay&apos;s  COPY  SWAP  repudiations marquee&apos;s kestrel&apos;s hallucinating Meighen&apos;s harm&apos;s Hershel wherefore dinginess&apos;s Uruguay&apos;s  ORIGINAL  repudiations marquee&apos;s kestrel&apos;s harm&apos;s Meighen&apos;s hallucinating Hershel wherefore dinginess&apos;s Uruguay&apos;s |  |

# 21.

Create a structure SkiLift with the following information:

* name(char array 15)
* maximum number of users for one ride
* whether the lift is in function.

Then create a structure SkiCenter with:

* name of the ski center (char array with max 20 character)
* the country(char array with max 20 chars)
* array of ski lifts(maximum 20 sifts)
* number of ski lifts.

Implement a function:

void biggestCapacity(SkiCenter \*sc, int n)

which will print the ski center with the biggest capacity for skiers at the same time. If there are two ski centers with the same capacity, print the one with more ski lifts. The printing should be in different lines (one for Name, one for Country, one for Capacity). When calculating the capacity, only consider the ski lifts which are in function!

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  Shapka  Makedonija  5  Teteks1  20  1  Teteks2  50  0  Ceripashina  100  0  Jelak  10  1  Aerodrom  36  1  Bansko  Bugarija  6  Tomba  40  0  Shiligarnika  80  1  Banderitsa  70  1  Todorka  36  1  ChalinValog  24  0  Plato  60  1 | Bansko  Bugarija  246 | Bansko  Bugarija  246 |  |
|  | 5  Kopaonik  Srbija  7  MaloJezero  30  1  Zlatiborac  80  1  PanchevVrh  80  0  Karaman  25  1  Jaram  40  1  Krcmar  120  1  Vucak  20  1  Borovets  Bugarija  8  Rila  18  1  Iglika  20  1  Rotata  36  1  Sitnyakovo  40  0  Martinovibaraki  60  0  Yastrebetz  80  1  Popangelov  100  1  Suhar  32  1  Mavrovo  Makedonija  6  Ednosed  30  1  Dvosed  60  1  Ciciban  10  1  Galicki  20  1  Belicki  36  0  Plavi  40  1  Shapka  Makedonija  5  Teteks1  20  1  Teteks2  50  0  Ceripashina  100  0  Jelak  10  1  Aerodrom  36  1  Bansko  Bugarija  6  Tomba  40  0  Shiligarnika  80  1  Banderitsa  70  1  Todorka  36  1  ChalinValog  24  0  Plato  60  1 | Kopaonik  Srbija  315 | Kopaonik  Srbija  315 |  |
|  | 1  Mavrovo  Makedonija  6  Ednosed  30  1  Dvosed  60  1  Ciciban  10  1  Galicki  20  1  Belicki  36  0  Plavi  40  1 | Mavrovo  Makedonija  160 | Mavrovo  Makedonija  160 |  |
|  | 3  Shapka  Makedonija  5  Teteks1  20  1  Teteks2  50  0  Ceripashina  100  0  Jelak  10  1  Aerodrom  36  1  Bansko  Bugarija  6  Tomba  40  0  Shiligarnika  80  1  Banderitsa  70  1  Todorka  36  1  ChalinValog  100  1  Plato  60  0  Borovets  Bugarija  8  Rila  18  1  Iglika  20  1  Rotata  36  1  Sitnyakovo  40  0  Martinovibaraki  60  0  Yastrebetz  80  1  Popangelov  100  1  Suhar  32  1 | Borovets  Bugarija  286 | Borovets  Bugarija  286 |  |
|  | 3  Shapka  Makedonija  5  Teteks1  20  1  Teteks2  50  0  Ceripashina  100  0  Jelak  10  1  Aerodrom  36  1  Bansko  Bugarija  6  Tomba  40  0  Shiligarnika  80  1  Banderitsa  70  1  Todorka  36  1  ChalinValog  56  1  Plato  60  1  Borovets  Bugarija  8  Rila  18  1  Iglika  20  1  Rotata  36  1  Sitnyakovo  40  0  Martinovibaraki  60  0  Yastrebetz  80  1  Popangelov  100  1  Suhar  32  1 | Bansko  Bugarija  302 | Bansko  Bugarija  302 |  |

# 22.

Implement a class String that keeps dynamically allocated array of characters and its length. For this class implement:

* appropriate constructors and destructor **(10 points)**
* operator [] for accessing character at given index i. If the index is less then 0 or greater then (the length of the array - 1) it should throw an exception of type IndexOutOfBounds. Handle this exception in the right place in the main function and print the message as in the sample output example. **(10 points)**

Implement class named StringBuilder that keeps dynamically allocated array of characters, the number of used characters and the total capacity. For this class implement:

* constructor StringBuilder(int capacity) that instantiate object of this class with given capacity and usage of 0 **(10 points)**
* operator += that accepts reference of the class String and adds the contents (the char array) from this class at the end of the character array of the object of the class StringBuilder. Therefore if there is not enough capacity, it should expand so:
  + if after doubling the capacity (capacity \* 2) there is enough capacity for the new array, the it's expanded in this manner
  + if after the expansion by doubling, still there is not enough space, the expand the array for what is needed to fit the new array you are adding
  + The original array should never be lost during the expansion. **(15 points)**
* operator << for printing objects of this class on the standard output in format: **(5 points)**

Capacity: {capacity}

Size: {size}

Contents: {contents}

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 34  44  wpfxmytkmjxvwjufgaqvolqabjbntqeivgkfjgem 30  eeqzoeva 32  tazajlzoxqvphqrcdgsnwmwwokuwjkuyoxmmczrxcddflcobqy 55  orczteqjgi 97  vrnpnliotqppquvlgzapisqyloxkn 29  urivtzlpfgtdtijdslfopwlbyemsgxvwzlmomveus 12  reijhnnoesxqbkzjjwtxlmibfj 70  qofimuinqdtdtjp 97  mgrqchxycueuqkxofgaskpdxavdgbkqdktjvzuqmz 47  vvhkpnvbgckdyjsxliz 57  xmsvtzgrfkbbjvnikcshsdhvdzoohneybijzsywlimtwiaeblf 4  slmflrqddlkpmdeaxdzygymhtjgrzun 74  uzzwliubgknqspjhidbebmovlqjssypgqcodiavdocrwrsgxd 83  qppmwlbrgfkvshahyrvxahjzjpiclgzjkskgtjpgjunn 77  eyotkqjcewbljvrslvfneesfenkbbvyn 81  usdppe -7  wounp -1  aizfbbrdkkzudlqxukguhiudnirenycxcznzzrxa 52  lfnydltjzvxnbnjgoyrrpftlebbqnuofslicbcvhjotgcwjcn 78  qdunldmnkwdorhyfymdnzehmekonfpmlpbqafhnerudgkb 2  gfkfrgjnyaze -9  grxrposnxgfbhxbvpmuqwwyqwopenyqf 80  hwmsyzulmbaubialmsuqpozaqodkjlqrysvpjq 75  pwruxilwbcyi 89  soqdlsofdpdxhwhbuoyyvnsk 44  yvbdbtdxcqnxhbbbzsvrvlskeahdjauugasg 55  kcxveblsfzugtfzxunmmschadenhwaggkonskmstpcfh -2  dhzchdinoqdxcnx 41  zgvqjvxbaihlbdjketheae 32  bobzoolyzubpkadcmbaljiysrnrefdxdd 72  deqcivorswovkcqtnyshnzkcldzyx 59  uuqjrpjilqynvwpuhvcwnewckcuywonemuwyu -9  ihfwhmqcqcnqixyzdqj 76  byrgrqryelszsjkvzkgty 12  fxiqdlfnwhnxwxsyibyjythqizxmogrnmgprsetoes 19  qlrgfogfzgjjcvvbtgxalixqgjlivzgsjoqccs 58  uxqhoimnmexjczgwijimyciwnrtkfxjekkqxvs 6  ywkqbfugtylyaxaefjktbhtnvn 22  fnztiiwioiqhmqkconett 0  wxplciefajgnktgzryoyvehcphrfdnhrcfljbmz 77  vadwoxhbauylnvfgrutklg 42  ezppuizpjt 65  mnvkgdwfquaypykfsqyehgtejvzblwxabwqhqnzdfz 42  rtyubulnejltrukhlemgdgewjufdhdywfwisgfip 79 | s[30] = e  s[32] = Index '32' is out of bounds  s[55] = Index '55' is out of bounds  s[97] = Index '97' is out of bounds  s[29] = Index '29' is out of bounds  s[12] = t  s[70] = Index '70' is out of bounds  s[97] = Index '97' is out of bounds  s[47] = Index '47' is out of bounds  s[57] = Index '57' is out of bounds  s[4] = t  s[74] = Index '74' is out of bounds  s[83] = Index '83' is out of bounds  s[77] = Index '77' is out of bounds  s[81] = Index '81' is out of bounds  s[-7] = Index '-7' is out of bounds  s[-1] = Index '-1' is out of bounds  s[52] = Index '52' is out of bounds  s[78] = Index '78' is out of bounds  s[2] = u  s[-9] = Index '-9' is out of bounds  s[80] = Index '80' is out of bounds  s[75] = Index '75' is out of bounds  s[89] = Index '89' is out of bounds  s[44] = Index '44' is out of bounds  s[55] = Index '55' is out of bounds  s[-2] = Index '-2' is out of bounds  s[41] = Index '41' is out of bounds  s[32] = Index '32' is out of bounds  s[72] = Index '72' is out of bounds  s[59] = Index '59' is out of bounds  s[-9] = Index '-9' is out of bounds  s[76] = Index '76' is out of bounds  s[12] = s  s[19] = j  s[58] = Index '58' is out of bounds  s[6] = m  s[22] = t  s[0] = f  s[77] = Index '77' is out of bounds  s[42] = Index '42' is out of bounds  s[65] = Index '65' is out of bounds  s[42] = Index '42' is out of bounds  s[79] = Index '79' is out of bounds  Capacity: 2176  Size: 1323  Contents: wpfxmytkmjxvwjufgaqvolqabjbntqeivgkfjgemeeqzoevatazajlzoxqvphqrcdgsnwmwwokuwjkuyoxmmczrxcddflcobqyorczteqjgivrnpnliotqppquvlgzapisqyloxknurivtzlpfgtdtijdslfopwlbyemsgxvwzlmomveusreijhnnoesxqbkzjjwtxlmibfjqofimuinqdtdtjpmgrqchxycueuqkxofgaskpdxavdgbkqdktjvzuqmzvvhkpnvbgckdyjsxlizxmsvtzgrfkbbjvnikcshsdhvdzoohneybijzsywlimtwiaeblfslmflrqddlkpmdeaxdzygymhtjgrzunuzzwliubgknqspjhidbebmovlqjssypgqcodiavdocrwrsgxdqppmwlbrgfkvshahyrvxahjzjpiclgzjkskgtjpgjunneyotkqjcewbljvrslvfneesfenkbbvynusdppewounpaizfbbrdkkzudlqxukguhiudnirenycxcznzzrxalfnydltjzvxnbnjgoyrrpftlebbqnuofslicbcvhjotgcwjcnqdunldmnkwdorhyfymdnzehmekonfpmlpbqafhnerudgkbgfkfrgjnyazegrxrposnxgfbhxbvpmuqwwyqwopenyqfhwmsyzulmbaubialmsuqpozaqodkjlqrysvpjqpwruxilwbcyisoqdlsofdpdxhwhbuoyyvnskyvbdbtdxcqnxhbbbzsvrvlskeahdjauugasgkcxveblsfzugtfzxunmmschadenhwaggkonskmstpcfhdhzchdinoqdxcnxzgvqjvxbaihlbdjketheaebobzoolyzubpkadcmbaljiysrnrefdxdddeqcivorswovkcqtnyshnzkcldzyxuuqjrpjilqynvwpuhvcwnewckcuywonemuwyuihfwhmqcqcnqixyzdqjbyrgrqryelszsjkvzkgtyfxiqdlfnwhnxwxsyibyjythqizxmogrnmgprsetoesqlrgfogfzgjjcvvbtgxalixqgjlivzgsjoqccsuxqhoimnmexjczgwijimyciwnrtkfxjekkqxvsywkqbfugtylyaxaefjktbhtnvnfnztiiwioiqhmqkconettwxplciefajgnktgzryoyvehcphrfdnhrcfljbmzvadwoxhbauylnvfgrutklgezppuizpjtmnvkgdwfquaypykfsqyehgtejvzblwxabwqhqnzdfzrtyubulnejltrukhlemgdgewjufdhdywfwisgfip | s[30] = e  s[32] = Index '32' is out of bounds  s[55] = Index '55' is out of bounds  s[97] = Index '97' is out of bounds  s[29] = Index '29' is out of bounds  s[12] = t  s[70] = Index '70' is out of bounds  s[97] = Index '97' is out of bounds  s[47] = Index '47' is out of bounds  s[57] = Index '57' is out of bounds  s[4] = t  s[74] = Index '74' is out of bounds  s[83] = Index '83' is out of bounds  s[77] = Index '77' is out of bounds  s[81] = Index '81' is out of bounds  s[-7] = Index '-7' is out of bounds  s[-1] = Index '-1' is out of bounds  s[52] = Index '52' is out of bounds  s[78] = Index '78' is out of bounds  s[2] = u  s[-9] = Index '-9' is out of bounds  s[80] = Index '80' is out of bounds  s[75] = Index '75' is out of bounds  s[89] = Index '89' is out of bounds  s[44] = Index '44' is out of bounds  s[55] = Index '55' is out of bounds  s[-2] = Index '-2' is out of bounds  s[41] = Index '41' is out of bounds  s[32] = Index '32' is out of bounds  s[72] = Index '72' is out of bounds  s[59] = Index '59' is out of bounds  s[-9] = Index '-9' is out of bounds  s[76] = Index '76' is out of bounds  s[12] = s  s[19] = j  s[58] = Index '58' is out of bounds  s[6] = m  s[22] = t  s[0] = f  s[77] = Index '77' is out of bounds  s[42] = Index '42' is out of bounds  s[65] = Index '65' is out of bounds  s[42] = Index '42' is out of bounds  s[79] = Index '79' is out of bounds  Capacity: 2176  Size: 1323  Contents: wpfxmytkmjxvwjufgaqvolqabjbntqeivgkfjgemeeqzoevatazajlzoxqvphqrcdgsnwmwwokuwjkuyoxmmczrxcddflcobqyorczteqjgivrnpnliotqppquvlgzapisqyloxknurivtzlpfgtdtijdslfopwlbyemsgxvwzlmomveusreijhnnoesxqbkzjjwtxlmibfjqofimuinqdtdtjpmgrqchxycueuqkxofgaskpdxavdgbkqdktjvzuqmzvvhkpnvbgckdyjsxlizxmsvtzgrfkbbjvnikcshsdhvdzoohneybijzsywlimtwiaeblfslmflrqddlkpmdeaxdzygymhtjgrzunuzzwliubgknqspjhidbebmovlqjssypgqcodiavdocrwrsgxdqppmwlbrgfkvshahyrvxahjzjpiclgzjkskgtjpgjunneyotkqjcewbljvrslvfneesfenkbbvynusdppewounpaizfbbrdkkzudlqxukguhiudnirenycxcznzzrxalfnydltjzvxnbnjgoyrrpftlebbqnuofslicbcvhjotgcwjcnqdunldmnkwdorhyfymdnzehmekonfpmlpbqafhnerudgkbgfkfrgjnyazegrxrposnxgfbhxbvpmuqwwyqwopenyqfhwmsyzulmbaubialmsuqpozaqodkjlqrysvpjqpwruxilwbcyisoqdlsofdpdxhwhbuoyyvnskyvbdbtdxcqnxhbbbzsvrvlskeahdjauugasgkcxveblsfzugtfzxunmmschadenhwaggkonskmstpcfhdhzchdinoqdxcnxzgvqjvxbaihlbdjketheaebobzoolyzubpkadcmbaljiysrnrefdxdddeqcivorswovkcqtnyshnzkcldzyxuuqjrpjilqynvwpuhvcwnewckcuywonemuwyuihfwhmqcqcnqixyzdqjbyrgrqryelszsjkvzkgtyfxiqdlfnwhnxwxsyibyjythqizxmogrnmgprsetoesqlrgfogfzgjjcvvbtgxalixqgjlivzgsjoqccsuxqhoimnmexjczgwijimyciwnrtkfxjekkqxvsywkqbfugtylyaxaefjktbhtnvnfnztiiwioiqhmqkconettwxplciefajgnktgzryoyvehcphrfdnhrcfljbmzvadwoxhbauylnvfgrutklgezppuizpjtmnvkgdwfquaypykfsqyehgtejvzblwxabwqhqnzdfzrtyubulnejltrukhlemgdgewjufdhdywfwisgfip |  |
|  | 40  42  auxpllcxgxhyxddouqruzlsdavnzmozbhofg 55  yrcwjqnnjpaeuxbyzyqyatbhoojpjhrure -1  celnyljmogyitiqpxxpvuaj 38  aqlavelxbkpcjrzmd 52  vdueunydwajxueahvwjpcvudmsusceduqchdeld 37  duzapvbxutzapehbichxfrryjuxwynmfamwrfciumbavqn 4  cuqatmvlizoyixzworucixerhznk 26  wdwmihrcq 38  ckxkmwgnzghfbxibeafvqgrivpftqptiqgmazmfxikonlkqax -10  smfrocmmrhedhsyqooomgyjvbaandfn 27  sadmodazvorbcs 41  kkhrelhbgugvnf 9  deujrcmloi 34  qpoluaqxmobashwxpqfvhrbsoqacvxqpjjrdsiwozg 66  gghlvxysaehmkjzwavykwifnuoyywoyczkqzip 9  hyxmzkdtilprvqaiummsm 31  srwokwdhdymtvtnpw -9  fjrdxmwlusykappcfbpgvvvm 70  manysxhongeulceuagnfekliqvkdiazv 5  uktsozwmfrwldrayucqh 64  jjxpbyku -1  ijkwrgddtcocsjahrqjwlpmrirnttcmmdeiubwtjf -3  fesnpqzgba -5  zveek 13  rwjqqmwwoi 36  fppevjwsaxsmtyfrdmqzwenoqopebovyexvllhbb 58  ntedugnqulebsogwfrgoknvyn 57  jxzfqlqlvgjachbvzpsfatvniawkt 61  bzofztqkwdmzrhxziqotfsjcmyaqdxcwtcsxuiwhazyjohhw 62  robnvluvyhjckksvrkonyeka 44  tjmgoaxdjwavvvvxaorltfgvpkakht 2  zfkcmqgzqayrybhzkylsdztwtrooxnthe 3  mypfujcnsekhvipdyefkgawzukinoqnetqkudt 41  mgeku 13  vdrrupddyonskjnunbmhbry 12  pxhfefgdcixtlofbjaleorfsboepdmlyhukys 15  cospcgrgxfw -7  xwlfzdbeliitotrzdcrtjzqsacpecw 17  fnptmzrbnbgcsqbeqbzvljxqwrflcwvzyxqualtfjt 70  cumjqdndlapqaisaydoncvouqgyakmowaxfphcuffyyxolhns 21  yseustkiibgpmmeedpemwstmhn 70  kwelbbzhvztuvyhduqwxl 29 | s[55] = Index '55' is out of bounds  s[-1] = Index '-1' is out of bounds  s[38] = Index '38' is out of bounds  s[52] = Index '52' is out of bounds  s[37] = l  s[4] = p  s[26] = n  s[38] = Index '38' is out of bounds  s[-10] = Index '-10' is out of bounds  s[27] = n  s[41] = Index '41' is out of bounds  s[9] = u  s[34] = Index '34' is out of bounds  s[66] = Index '66' is out of bounds  s[9] = e  s[31] = Index '31' is out of bounds  s[-9] = Index '-9' is out of bounds  s[70] = Index '70' is out of bounds  s[5] = x  s[64] = Index '64' is out of bounds  s[-1] = Index '-1' is out of bounds  s[-3] = Index '-3' is out of bounds  s[-5] = Index '-5' is out of bounds  s[13] = Index '13' is out of bounds  s[36] = Index '36' is out of bounds  s[58] = Index '58' is out of bounds  s[57] = Index '57' is out of bounds  s[61] = Index '61' is out of bounds  s[62] = Index '62' is out of bounds  s[44] = Index '44' is out of bounds  s[2] = m  s[3] = c  s[41] = Index '41' is out of bounds  s[13] = Index '13' is out of bounds  s[12] = k  s[15] = b  s[-7] = Index '-7' is out of bounds  s[17] = c  s[70] = Index '70' is out of bounds  s[21] = v  s[70] = Index '70' is out of bounds  s[29] = Index '29' is out of bounds  Capacity: 1280  Size: 1129  Contents: auxpllcxgxhyxddouqruzlsdavnzmozbhofgyrcwjqnnjpaeuxbyzyqyatbhoojpjhrurecelnyljmogyitiqpxxpvuajaqlavelxbkpcjrzmdvdueunydwajxueahvwjpcvudmsusceduqchdeldduzapvbxutzapehbichxfrryjuxwynmfamwrfciumbavqncuqatmvlizoyixzworucixerhznkwdwmihrcqckxkmwgnzghfbxibeafvqgrivpftqptiqgmazmfxikonlkqaxsmfrocmmrhedhsyqooomgyjvbaandfnsadmodazvorbcskkhrelhbgugvnfdeujrcmloiqpoluaqxmobashwxpqfvhrbsoqacvxqpjjrdsiwozggghlvxysaehmkjzwavykwifnuoyywoyczkqziphyxmzkdtilprvqaiummsmsrwokwdhdymtvtnpwfjrdxmwlusykappcfbpgvvvmmanysxhongeulceuagnfekliqvkdiazvuktsozwmfrwldrayucqhjjxpbykuijkwrgddtcocsjahrqjwlpmrirnttcmmdeiubwtjffesnpqzgbazveekrwjqqmwwoifppevjwsaxsmtyfrdmqzwenoqopebovyexvllhbbntedugnqulebsogwfrgoknvynjxzfqlqlvgjachbvzpsfatvniawktbzofztqkwdmzrhxziqotfsjcmyaqdxcwtcsxuiwhazyjohhwrobnvluvyhjckksvrkonyekatjmgoaxdjwavvvvxaorltfgvpkakhtzfkcmqgzqayrybhzkylsdztwtrooxnthemypfujcnsekhvipdyefkgawzukinoqnetqkudtmgekuvdrrupddyonskjnunbmhbrypxhfefgdcixtlofbjaleorfsboepdmlyhukyscospcgrgxfwxwlfzdbeliitotrzdcrtjzqsacpecwfnptmzrbnbgcsqbeqbzvljxqwrflcwvzyxqualtfjtcumjqdndlapqaisaydoncvouqgyakmowaxfphcuffyyxolhnsyseustkiibgpmmeedpemwstmhnkwelbbzhvztuvyhduqwxl | s[55] = Index '55' is out of bounds  s[-1] = Index '-1' is out of bounds  s[38] = Index '38' is out of bounds  s[52] = Index '52' is out of bounds  s[37] = l  s[4] = p  s[26] = n  s[38] = Index '38' is out of bounds  s[-10] = Index '-10' is out of bounds  s[27] = n  s[41] = Index '41' is out of bounds  s[9] = u  s[34] = Index '34' is out of bounds  s[66] = Index '66' is out of bounds  s[9] = e  s[31] = Index '31' is out of bounds  s[-9] = Index '-9' is out of bounds  s[70] = Index '70' is out of bounds  s[5] = x  s[64] = Index '64' is out of bounds  s[-1] = Index '-1' is out of bounds  s[-3] = Index '-3' is out of bounds  s[-5] = Index '-5' is out of bounds  s[13] = Index '13' is out of bounds  s[36] = Index '36' is out of bounds  s[58] = Index '58' is out of bounds  s[57] = Index '57' is out of bounds  s[61] = Index '61' is out of bounds  s[62] = Index '62' is out of bounds  s[44] = Index '44' is out of bounds  s[2] = m  s[3] = c  s[41] = Index '41' is out of bounds  s[13] = Index '13' is out of bounds  s[12] = k  s[15] = b  s[-7] = Index '-7' is out of bounds  s[17] = c  s[70] = Index '70' is out of bounds  s[21] = v  s[70] = Index '70' is out of bounds  s[29] = Index '29' is out of bounds  Capacity: 1280  Size: 1129  Contents: auxpllcxgxhyxddouqruzlsdavnzmozbhofgyrcwjqnnjpaeuxbyzyqyatbhoojpjhrurecelnyljmogyitiqpxxpvuajaqlavelxbkpcjrzmdvdueunydwajxueahvwjpcvudmsusceduqchdeldduzapvbxutzapehbichxfrryjuxwynmfamwrfciumbavqncuqatmvlizoyixzworucixerhznkwdwmihrcqckxkmwgnzghfbxibeafvqgrivpftqptiqgmazmfxikonlkqaxsmfrocmmrhedhsyqooomgyjvbaandfnsadmodazvorbcskkhrelhbgugvnfdeujrcmloiqpoluaqxmobashwxpqfvhrbsoqacvxqpjjrdsiwozggghlvxysaehmkjzwavykwifnuoyywoyczkqziphyxmzkdtilprvqaiummsmsrwokwdhdymtvtnpwfjrdxmwlusykappcfbpgvvvmmanysxhongeulceuagnfekliqvkdiazvuktsozwmfrwldrayucqhjjxpbykuijkwrgddtcocsjahrqjwlpmrirnttcmmdeiubwtjffesnpqzgbazveekrwjqqmwwoifppevjwsaxsmtyfrdmqzwenoqopebovyexvllhbbntedugnqulebsogwfrgoknvynjxzfqlqlvgjachbvzpsfatvniawktbzofztqkwdmzrhxziqotfsjcmyaqdxcwtcsxuiwhazyjohhwrobnvluvyhjckksvrkonyekatjmgoaxdjwavvvvxaorltfgvpkakhtzfkcmqgzqayrybhzkylsdztwtrooxnthemypfujcnsekhvipdyefkgawzukinoqnetqkudtmgekuvdrrupddyonskjnunbmhbrypxhfefgdcixtlofbjaleorfsboepdmlyhukyscospcgrgxfwxwlfzdbeliitotrzdcrtjzqsacpecwfnptmzrbnbgcsqbeqbzvljxqwrflcwvzyxqualtfjtcumjqdndlapqaisaydoncvouqgyakmowaxfphcuffyyxolhnsyseustkiibgpmmeedpemwstmhnkwelbbzhvztuvyhduqwxl |  |
|  | 29  22  dhwnlppwbsu -4  ofpwutwlbshckexuzklfssglkqfnbfunvwgibhnzrt 46  xilzptrwtzreokldwcfhutyvkahwjyx 58  wpvcfyfljpii 59  lbvbwx 43  bfgdoleiwhciiqxkvlqreabtumux 47  yfdkooed -5  qoamaqjzdfiqmgrlffef 15  wuqidhlxighwynohfnacnjov 13  gjktdwwquirathlljiqjviscwhedwcuennvrzgseaplskvu 49  bkzhmwvevmwpcijtiithphldd 62  bczyqyiroqpymhk 12  vbgmfpqjmkniqtgunpgnigbdtudyszsieghpgxwpod 15  gnunwdjlcwdosfp -4  eggijpcnxllejros 54  tlerlpspdrpqtr 54  ayubywcujbykhgfawgjz 62  uwixvbybzuwwbatjousebaamflrr 27  tkswnvepuagptbqgelcfulmp 4  tpvpmgnyyqqmehuvyrikdlfwrv 26  kkxjjyglnhjummodnzreefzntoklbeawbazdeyhlachgpiqm 30  yarbopwrqmfagzfodebzguqrwtqrjyeguf 37 | s[-4] = Index '-4' is out of bounds  s[46] = Index '46' is out of bounds  s[58] = Index '58' is out of bounds  s[59] = Index '59' is out of bounds  s[43] = Index '43' is out of bounds  s[47] = Index '47' is out of bounds  s[-5] = Index '-5' is out of bounds  s[15] = l  s[13] = n  s[49] = Index '49' is out of bounds  s[62] = Index '62' is out of bounds  s[12] = m  s[15] = u  s[-4] = Index '-4' is out of bounds  s[54] = Index '54' is out of bounds  s[54] = Index '54' is out of bounds  s[62] = Index '62' is out of bounds  s[27] = r  s[4] = n  s[26] = Index '26' is out of bounds  s[30] = a  s[37] = Index '37' is out of bounds  Capacity: 928  Size: 536  Contents: dhwnlppwbsuofpwutwlbshckexuzklfssglkqfnbfunvwgibhnzrtxilzptrwtzreokldwcfhutyvkahwjyxwpvcfyfljpiilbvbwxbfgdoleiwhciiqxkvlqreabtumuxyfdkooedqoamaqjzdfiqmgrlffefwuqidhlxighwynohfnacnjovgjktdwwquirathlljiqjviscwhedwcuennvrzgseaplskvubkzhmwvevmwpcijtiithphlddbczyqyiroqpymhkvbgmfpqjmkniqtgunpgnigbdtudyszsieghpgxwpodgnunwdjlcwdosfpeggijpcnxllejrostlerlpspdrpqtrayubywcujbykhgfawgjzuwixvbybzuwwbatjousebaamflrrtkswnvepuagptbqgelcfulmptpvpmgnyyqqmehuvyrikdlfwrvkkxjjyglnhjummodnzreefzntoklbeawbazdeyhlachgpiqmyarbopwrqmfagzfodebzguqrwtqrjyeguf | s[-4] = Index '-4' is out of bounds  s[46] = Index '46' is out of bounds  s[58] = Index '58' is out of bounds  s[59] = Index '59' is out of bounds  s[43] = Index '43' is out of bounds  s[47] = Index '47' is out of bounds  s[-5] = Index '-5' is out of bounds  s[15] = l  s[13] = n  s[49] = Index '49' is out of bounds  s[62] = Index '62' is out of bounds  s[12] = m  s[15] = u  s[-4] = Index '-4' is out of bounds  s[54] = Index '54' is out of bounds  s[54] = Index '54' is out of bounds  s[62] = Index '62' is out of bounds  s[27] = r  s[4] = n  s[26] = Index '26' is out of bounds  s[30] = a  s[37] = Index '37' is out of bounds  Capacity: 928  Size: 536  Contents: dhwnlppwbsuofpwutwlbshckexuzklfssglkqfnbfunvwgibhnzrtxilzptrwtzreokldwcfhutyvkahwjyxwpvcfyfljpiilbvbwxbfgdoleiwhciiqxkvlqreabtumuxyfdkooedqoamaqjzdfiqmgrlffefwuqidhlxighwynohfnacnjovgjktdwwquirathlljiqjviscwhedwcuennvrzgseaplskvubkzhmwvevmwpcijtiithphlddbczyqyiroqpymhkvbgmfpqjmkniqtgunpgnigbdtudyszsieghpgxwpodgnunwdjlcwdosfpeggijpcnxllejrostlerlpspdrpqtrayubywcujbykhgfawgjzuwixvbybzuwwbatjousebaamflrrtkswnvepuagptbqgelcfulmptpvpmgnyyqqmehuvyrikdlfwrvkkxjjyglnhjummodnzreefzntoklbeawbazdeyhlachgpiqmyarbopwrqmfagzfodebzguqrwtqrjyeguf |  |
|  | 10  5  Objektno 4  Orientirano 78  Programiranje 11  FINKI 0  2015 3 | s[4] = k  s[78] = Index '78' is out of bounds  s[11] = j  s[0] = F  s[3] = 5  Capacity: 84  Size: 41  Contents: ObjektnoOrientiranoProgramiranjeFINKI2015 | s[4] = k  s[78] = Index '78' is out of bounds  s[11] = j  s[0] = F  s[3] = 5  Capacity: 84  Size: 41  Contents: ObjektnoOrientiranoProgramiranjeFINKI2015 |  |
|  | 48  36  jpvjfuhdcffpzpuihglxuyfpxpnoaio 17  sbvhdcdsxiryulnrfikrzdkajqwuubbacsnuyl 12  bmdqtdqrlonhznbpahdupprmgpswbcplkkdrolntn 16  fruixwjrmnwjuzgdofjbehuenqa 69  gbexfbv 45  wvuuwtfpzifhljcliuobjqnanhg 53  ngqvoanvefezfpmpilaltpwrztnxcxlpmpdnsstgyqi -3  fmkemhgitzlijidgrjzmzmuuawmwfgrdxtnypf 31  lkwlfojfecwlflkuhosighhxqx 18  hrwdiqvbumivuyhobaapzsxccuvnicbfngwmhpbrsnvorn 1  yhpkdhrewfretgtouuajyzolloozztiqccyeloitmj 21  awlujeojwretimllfcndypjurhjkwak 55  xjqfujruwqqluqqqqcbsepcwf 69  okrmbkblaekrrjnvpebfauq 22  imwevnhfvnvfvubbk 19  lznjrhvcfszlpcxyctxwamevjoottawrnxixtcpm -6  lnqezqqtzmo 40  puyrebtrfdexzkukybcmdwb 48  wcemqjjaw 8  lygzanvuovsdxfzhdrcrxdipwlqaezrh 13  hixasxymnv 39  lugahoiwwcyjhitldbqckltzlmccbcbfa 24  iebfk 50  ecsuwmfzocymwbbwyopmzcmugrtjskshwtwvs 49  dswmlmbveavyasjbezqlkucamvrezlh 30  msuwyfwaijvuklhpcpnxaegerk -1  rbuogluacmjznaupxrlczxvgfftzsrdkhpwmaypbekuruf 43  xhtdtyehrfvbmnszu 29  iyhtopfkqalewpboqqlebwgx 1  puhzobbdxhcxmdplclibzdwcb 45  pwrwphgegwpmsdknv -6  dpelwncmxglmczrkogepsiqmvjmdprfwdywpjn 65  fusjgwoccmimwszmrznpwblrnmtlnsqpkmolsoifpkurv 16  rzotcdbeaktwejpomj 16  fpefioclznzv 36  zxoobjbxowaaspoyreldvmmkltqmnnqyluhzdyurctnlcalfu 33 | s[17] = g  s[12] = u  s[16] = a  s[69] = Index '69' is out of bounds  s[45] = Index '45' is out of bounds  s[53] = Index '53' is out of bounds  s[-3] = Index '-3' is out of bounds  s[31] = d  s[18] = s  s[1] = r  s[21] = z  s[55] = Index '55' is out of bounds  s[69] = Index '69' is out of bounds  s[22] = q  s[19] = Index '19' is out of bounds  s[-6] = Index '-6' is out of bounds  s[40] = Index '40' is out of bounds  s[48] = Index '48' is out of bounds  s[8] = w  s[13] = f  s[39] = Index '39' is out of bounds  s[24] = l  s[50] = Index '50' is out of bounds  s[49] = Index '49' is out of bounds  s[30] = h  s[-1] = Index '-1' is out of bounds  s[43] = r  s[29] = Index '29' is out of bounds  s[1] = y  s[45] = Index '45' is out of bounds  s[-6] = Index '-6' is out of bounds  s[65] = Index '65' is out of bounds  s[16] = r  s[16] = m  s[36] = Index '36' is out of bounds  s[33] = u  Capacity: 1536  Size: 1010  Contents: jpvjfuhdcffpzpuihglxuyfpxpnoaiosbvhdcdsxiryulnrfikrzdkajqwuubbacsnuylbmdqtdqrlonhznbpahdupprmgpswbcplkkdrolntnfruixwjrmnwjuzgdofjbehuenqagbexfbvwvuuwtfpzifhljcliuobjqnanhgngqvoanvefezfpmpilaltpwrztnxcxlpmpdnsstgyqifmkemhgitzlijidgrjzmzmuuawmwfgrdxtnypflkwlfojfecwlflkuhosighhxqxhrwdiqvbumivuyhobaapzsxccuvnicbfngwmhpbrsnvornyhpkdhrewfretgtouuajyzolloozztiqccyeloitmjawlujeojwretimllfcndypjurhjkwakxjqfujruwqqluqqqqcbsepcwfokrmbkblaekrrjnvpebfauqimwevnhfvnvfvubbklznjrhvcfszlpcxyctxwamevjoottawrnxixtcpmlnqezqqtzmopuyrebtrfdexzkukybcmdwbwcemqjjawlygzanvuovsdxfzhdrcrxdipwlqaezrhhixasxymnvlugahoiwwcyjhitldbqckltzlmccbcbfaiebfkecsuwmfzocymwbbwyopmzcmugrtjskshwtwvsdswmlmbveavyasjbezqlkucamvrezlhmsuwyfwaijvuklhpcpnxaegerkrbuogluacmjznaupxrlczxvgfftzsrdkhpwmaypbekurufxhtdtyehrfvbmnszuiyhtopfkqalewpboqqlebwgxpuhzobbdxhcxmdplclibzdwcbpwrwphgegwpmsdknvdpelwncmxglmczrkogepsiqmvjmdprfwdywpjnfusjgwoccmimwszmrznpwblrnmtlnsqpkmolsoifpkurvrzotcdbeaktwejpomjfpefioclznzvzxoobjbxowaaspoyreldvmmkltqmnnqyluhzdyurctnlcalfu | s[17] = g  s[12] = u  s[16] = a  s[69] = Index '69' is out of bounds  s[45] = Index '45' is out of bounds  s[53] = Index '53' is out of bounds  s[-3] = Index '-3' is out of bounds  s[31] = d  s[18] = s  s[1] = r  s[21] = z  s[55] = Index '55' is out of bounds  s[69] = Index '69' is out of bounds  s[22] = q  s[19] = Index '19' is out of bounds  s[-6] = Index '-6' is out of bounds  s[40] = Index '40' is out of bounds  s[48] = Index '48' is out of bounds  s[8] = w  s[13] = f  s[39] = Index '39' is out of bounds  s[24] = l  s[50] = Index '50' is out of bounds  s[49] = Index '49' is out of bounds  s[30] = h  s[-1] = Index '-1' is out of bounds  s[43] = r  s[29] = Index '29' is out of bounds  s[1] = y  s[45] = Index '45' is out of bounds  s[-6] = Index '-6' is out of bounds  s[65] = Index '65' is out of bounds  s[16] = r  s[16] = m  s[36] = Index '36' is out of bounds  s[33] = u  Capacity: 1536  Size: 1010  Contents: jpvjfuhdcffpzpuihglxuyfpxpnoaiosbvhdcdsxiryulnrfikrzdkajqwuubbacsnuylbmdqtdqrlonhznbpahdupprmgpswbcplkkdrolntnfruixwjrmnwjuzgdofjbehuenqagbexfbvwvuuwtfpzifhljcliuobjqnanhgngqvoanvefezfpmpilaltpwrztnxcxlpmpdnsstgyqifmkemhgitzlijidgrjzmzmuuawmwfgrdxtnypflkwlfojfecwlflkuhosighhxqxhrwdiqvbumivuyhobaapzsxccuvnicbfngwmhpbrsnvornyhpkdhrewfretgtouuajyzolloozztiqccyeloitmjawlujeojwretimllfcndypjurhjkwakxjqfujruwqqluqqqqcbsepcwfokrmbkblaekrrjnvpebfauqimwevnhfvnvfvubbklznjrhvcfszlpcxyctxwamevjoottawrnxixtcpmlnqezqqtzmopuyrebtrfdexzkukybcmdwbwcemqjjawlygzanvuovsdxfzhdrcrxdipwlqaezrhhixasxymnvlugahoiwwcyjhitldbqckltzlmccbcbfaiebfkecsuwmfzocymwbbwyopmzcmugrtjskshwtwvsdswmlmbveavyasjbezqlkucamvrezlhmsuwyfwaijvuklhpcpnxaegerkrbuogluacmjznaupxrlczxvgfftzsrdkhpwmaypbekurufxhtdtyehrfvbmnszuiyhtopfkqalewpboqqlebwgxpuhzobbdxhcxmdplclibzdwcbpwrwphgegwpmsdknvdpelwncmxglmczrkogepsiqmvjmdprfwdywpjnfusjgwoccmimwszmrznpwblrnmtlnsqpkmolsoifpkurvrzotcdbeaktwejpomjfpefioclznzvzxoobjbxowaaspoyreldvmmkltqmnnqyluhzdyurctnlcalfu |  |

Passed

# 23.

For the needs of an Internet operator, it is necessary to model a system for bill (invoice) payment.

For that purpose, create an abstract class for representing a monthly invoice (MonthlyInvoice) of an Internet operator. Each invoice is represented by the following information:

* user name (array of up to 100 characters)
* user number (integer)
* initial price of the package (float number)
* amount of used internet in a given month in gigabytes (integer)

Implement separate derived classes for different types of invoices: for a business user (BusinessInvoice) and a private user (PrivateInvoice).

Each invoice for a business user is represented by the following information:

* whether the line is optical or cable (boolean variable, optics - true, cable -false)
* quantity of included free transfer (in gigabytes - int)

Each invoice for a private user is represented by the following information:

* identification number of the citizen (array of up to 13 characters)
* dynamically allocated array of of the sizes of online movies from the video collection. The array keeps an information of the size (in GB) of each movie that was watched by the private user within the given month. If the user watched 3 movies from the video, then 3 integers are stored to represent the size in gigabytes of each movie (eg. the {1, 2, 4} array represents 1GB, 2GB and 4GB movies respectively.
* number of movies watched from the video collection (integer).

Calculate the total cost of the invoice for each of the derived classes.

* For a business user it is calculated as: initial\_price + amount\_of\_used\_internet\_over\_the\_included\_free\_internet \* 100. If the connection is optical, the initial price increases by 50%.
* For a private user, the price is counted as: 'initial*price + amount*of*used*internet \* 50 + total*amount*of*gigabytes*from*the*watched*movies*from*the*movie\_collection \* 10`.

Define the following functionalities for the BusinessInvoice:

* *\* operator ++ (post notation) \** - increases the amount of the used internet by 1GB

Define the following operators for the PrivateInvoice class:

* *\* operator [] \** - returns the size of the movie on a given position specified by the input argument. If you search for a movie on an index that does not exist, it returns -1 (for example: if there are 3 movies and the element with an index 5 is required, it should return -1). The first film is on position (index) '0'.
* *\* operator += \** - adds a new movie size in the array of watched movies
* *\* operator ++ \** - increases the amount of the used internet by 1GB

Overload the operator > for comparison of two accounts based on the total cost of their monthly invoice. Overload the << operator that will print all the relevant information for each user in the following format:

* For the business invoice:
* [user number] - [user name], [initial price]
* [line type], [quantity of the used internet over the free gigabytes]
* For the private invoice:
* [user number] - [user name], [initial price]
* [used gigabytes]
* [number of watched movies], [total movie size (in gigabytes)]

Write the `biggetsInvoice ' function that accepts an array of pointers to the invoices as input argument and returns an object (of the appropriate type) that represents the invoice with the highest total cost.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  Finki  123456  20000  1000  1  1999 | Testing total\_price()  123456 - Finki, 20000  optical link, 0  30000  123456 - Finki, 20000  optical link, 0  30000 | Testing total\_price()  123456 - Finki, 20000  optical link, 0  30000  123456 - Finki, 20000  optical link, 0  30000 |  |
|  | 3  Finki  123456  20000  1999  1  1000 | Testing total\_price()  123456 - Finki, 20000  optical link, 999  129900  123456 - Finki, 20000  optical link, 1000  130000 | Testing total\_price()  123456 - Finki, 20000  optical link, 999  129900  123456 - Finki, 20000  optical link, 1000  130000 |  |
|  | 6  Petar  987  500  50  4  1  2  3  4 | 987 - Petar, 500  50  4, 10 | 987 - Petar, 500  50  4, 10 |  |
|  | 7  Petar  987  500  50  4  1  2  1  2 | 987 - Petar, 500  50  4, 6  Movie sizes: [1 2 1 2 ] | 987 - Petar, 500  50  4, 6  Movie sizes: [1 2 1 2 ] |  |
|  | 5  Petar  987  500  50  5 | 987 - Petar, 500  55  0, 0 | 987 - Petar, 500  55  0, 0 |  |
|  | 3  Finki  123456  20000  1000  0  1000 | Testing total\_price()  123456 - Finki, 20000  cable link, 0  20000  123456 - Finki, 20000  cable link, 1  20100 | Testing total\_price()  123456 - Finki, 20000  cable link, 0  20000  123456 - Finki, 20000  cable link, 1  20100 |  |
|  | 9  2  1  Finki  1234  10000  2059  1  2000  100  2  Nikola  999  1000  100  1  5  10 | 1234 - Finki, 10000  optical link, 159  30900  999 - Nikola, 1000  110  1, 5  6550  The biggest invoice is:  1234 - Finki, 10000  optical link, 159  30900 | 1234 - Finki, 10000  optical link, 159  30900  999 - Nikola, 1000  110  1, 5  6550  The biggest invoice is:  1234 - Finki, 10000  optical link, 159  30900 |  |
|  | 2  Finki  123456  20000  1000  1  500  50 | BusinessInvoice:  123456 - Finki, 20000  optical link, 500  testing operator++  123456 - Finki, 20000  optical link, 550 | BusinessInvoice:  123456 - Finki, 20000  optical link, 500  testing operator++  123456 - Finki, 20000  optical link, 550 |  |
|  | 2  Finki  123456  20000  1000  0  1000  3 | BusinessInvoice:  123456 - Finki, 20000  cable link, 0  testing operator++  123456 - Finki, 20000  cable link, 3 | BusinessInvoice:  123456 - Finki, 20000  cable link, 0  testing operator++  123456 - Finki, 20000  cable link, 3 |  |
|  | 3  Finki  123456  20000  500  1  1000 | Testing total\_price()  123456 - Finki, 20000  optical link, 0  30000  123456 - Finki, 20000  optical link, 0  30000 | Testing total\_price()  123456 - Finki, 20000  optical link, 0  30000  123456 - Finki, 20000  optical link, 0  30000 |  |
|  | 2  Finki  123456  20000  500  1  1000  500 | BusinessInvoice:  123456 - Finki, 20000  optical link, 0  testing operator++  123456 - Finki, 20000  optical link, 0 | BusinessInvoice:  123456 - Finki, 20000  optical link, 0  testing operator++  123456 - Finki, 20000  optical link, 0 |  |
|  | 4  Petar  987  500  50 | Private Invoice Created:  987 - Petar, 500  50  0, 0 | Private Invoice Created:  987 - Petar, 500  50  0, 0 |  |
|  | 8  Petar  987  500  50  4  1  2  3  4  20 | 987 - Petar, 500  50  0, 0  3000  987 - Petar, 500  70  4, 10  4100 | 987 - Petar, 500  50  0, 0  3000  987 - Petar, 500  70  4, 10  4100 |  |
|  | 1  Finki  123456  2000  0  1  10000 | BusinessInvoice Created:  123456 - Finki, 2000  optical link, 0 | BusinessInvoice Created:  123456 - Finki, 2000  optical link, 0 |  |
|  | 9  2  2  Petar  987  500  50  4  1  2  3  4  9  2  Nikola  999  1000  100  1  5  10 | 987 - Petar, 500  59  4, 10  3550  999 - Nikola, 1000  110  1, 5  6550  The biggest invoice is:  999 - Nikola, 1000  110  1, 5  6550 | 987 - Petar, 500  59  4, 10  3550  999 - Nikola, 1000  110  1, 5  6550  The biggest invoice is:  999 - Nikola, 1000  110  1, 5  6550 |  |

Passed

# 24.

Create a Race class. Each race is represented by the following information:

* the name of the city in which it is performed (array of up to 40 characters)
* year (integer)
* month (integer)
* best time in seconds (float)
* length in km (float)

For this class, implement the following methods:

* method complexity() that calculates the complexity of the race according to the following formula:

best time in seconds / length in km

* operator << for printing the race in the following format:

[City] [month] [year] [complexity]

Create a MotoGPRace class that, besides with the basic information about the race, additionally is represented by the following information:

* array of best times of preparation (dynamically allocated array of float numbers)
* the number of best times (integer)
* number of laps (integer)

In this class, override the complexity() method so that the average of the best times of the preparations multiplied by the MotoGP coefficient will be added to the initial complexity of the race. MotoGP coefficient **is the same for all** MotoGP races and can be changed by the commission (the initial value is 0.4). Additionally, if the number of laps is greater than 22, the complexity is increased by 20%.

For this class, provide the following methods:

* operator += - to add a new time of preparation to the dynamically allocated array of the best times of preparation. If an attempt is made to add a value less than 9.5, throw an exception (an object of the 'InvalidTimeException' class). The exception handling should be performed in the main function on the appropriate place. If an exception was thrown, an appropriate error message "**Invalid Time**" should be printed, and this value should not be entered in the array.
* Operator ++ in postfix notation - to increase the number of laps by one for a given MotoGpRace.

Provide all the necessary functions for the proper functioning of the program.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5  5  1  Madrid  2015  1  206.3  34.5  2  Mugello  2019  6  99.7  5.245  20  4  105.2  103.4  99.99  34.2  1  Skopje  2013  8  70.0  1844.5  2  Brno  2017  7  85.824  4.533  24  4  105.2  103.4  99.99  34.2  1  Sofia  2015  9  85.824  4.533  55.55 | ===== Testing - coefficient ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 53.2876  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 78.9968  Skopje 8.2013 0.0379507  Brno 7.2017 94.7057  Sofia 9.2015 18.9332 | ===== Testing - coefficient ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 53.2876  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 78.9968  Skopje 8.2013 0.0379507  Brno 7.2017 94.7057  Sofia 9.2015 18.9332 |  |
|  | 4  Valencia  2012  4  59.424  4.122  21  4  105.2  103.4  99.99  34.2  33.3  8.5 | ===== Testing - operator++ ======  21  Valencia 4.2012 48.6953  Valencia 4.2012 58.4344 | ===== Testing - operator++ ======  21  Valencia 4.2012 48.6953  Valencia 4.2012 58.4344 |  |
|  | 1  Paris  2011  5  1986.3  49.6  Jerez  2017  6  41.53  4.429  24  5  49.2  39.11  42.4  54.3  44.4 | ===== Testing - classes ======  Paris 5.2011 40.0464  Jerez 6.2017 33.2756 | ===== Testing - classes ======  Paris 5.2011 40.0464  Jerez 6.2017 33.2756 |  |
|  | 3  Qatar  2011  3  54.822  5.535  24  33.33  5.5 | ===== Testing - exceptions ======  Invalid Time  Qatar 3.2011 27.8839 | ===== Testing - exceptions ======  Invalid Time  Qatar 3.2011 27.8839 |  |
|  | 2  5  1  Madrid  2015  1  206.3  34.5  2  Mugello  2019  6  99.7  5.245  20  4  105.2  103.4  99.99  34.2  1  Skopje  2013  8  70.0  1844.5  2  Brno  2017  7  85.824  4.533  24  4  105.2  103.4  99.99  34.2  1  Sofia  2015  9  85.824  4.533  55.55 | ===== Testing - operatorot += ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 53.2876  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332  Lista na site Trki:  Madrid 1.2015 5.97971  Mugello 6.2019 50.8758  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332 | ===== Testing - operatorot += ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 53.2876  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332  Lista na site Trki:  Madrid 1.2015 5.97971  Mugello 6.2019 50.8758  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332 |  |
|  | 6  7  1  Madrid  2015  1  206.3  34.5  2  Mugello  2019  6  99.7  5.245  20  4  105.2  103.4  99.99  34.2  1  Skopje  2013  8  70.0  1844.5  2  Brno  2017  7  85.824  4.533  24  4  105.2  103.4  99.99  34.2  1  Sofia  2015  9  85.824  4.533  2  Qatar  2016  5  75.7  7.444  21  6  78.212  66.33  87.789  59.1  111.2  123.3  2  Bankog  2009  10  55.7  6.664  20  6  55.212  54.33  56.789  57.1  59.2  51.3  0.3  55.4 | ===== Testing - full functionality ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 53.2876  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332  Qatar 5.2016 45.2313  Bankog 10.2009 30.6204  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 44.7178  Skopje 8.2013 0.0379507  Brno 7.2017 53.5709  Sofia 9.2015 18.9332  Qatar 5.2016 36.4658  Bankog 10.2009 25.0549  Invalid Time  List of the races:  Madrid 1.2015 5.97971  Mugello 6.2019 42.9  Skopje 8.2013 0.0379507  Brno 7.2017 51.3895  Sofia 9.2015 18.9332  Qatar 5.2016 35.0835  Bankog 10.2009 25.044 | ===== Testing - full functionality ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 53.2876  Skopje 8.2013 0.0379507  Brno 7.2017 63.8546  Sofia 9.2015 18.9332  Qatar 5.2016 45.2313  Bankog 10.2009 30.6204  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 44.7178  Skopje 8.2013 0.0379507  Brno 7.2017 53.5709  Sofia 9.2015 18.9332  Qatar 5.2016 36.4658  Bankog 10.2009 25.0549  Invalid Time  List of the races:  Madrid 1.2015 5.97971  Mugello 6.2019 42.9  Skopje 8.2013 0.0379507  Brno 7.2017 51.3895  Sofia 9.2015 18.9332  Qatar 5.2016 35.0835  Bankog 10.2009 25.044 |  |

Passed

# 25.

Create class Tickets that has info:

* ID (array 12 digits),
* event (char array max 100)
* base price (decimal number)

The ticket store offers two types of tickets that should be defined in two separate classes: Online and Offline. For the Online type keep additional data if the member who byes the ticket is premium user or not, and for the second type Offline additionally keep if it has reservation for seat or not.

For the classes implement:

* needed constructors and destructor (see the calls in the main function) **(5 points)**
* operator << for printing the tickets in format: **(5 points)**

ID - event - computed price

* method bool falsifikat(int from, int to) for checking if the ticket is fake. You should check if the last 4 digits of the ID of the ticket are in the range [from-to] **(10 points)**
* method float presmetajCena() for computing the price of the separate types of tickets on the following way: **(10 points)**
  + for Оnline - price is decreased for 15% of the basic price if the memeber is premium
  + for Offline - price is increased for 10% of the basic price if there is seat reservation

Implemented additional functions with signature void vkupenPrihod(Tickets \*\* t, int n, int from, int to) that will print the total income of the tickets that are not fake. **(10 points)**

In the class Tickets there is integer that is shared between all tickets and it is the discount of the basic price for all objects. Initial value of this variable is 0. For manipulation of this variable implement the following static method: **(5 pionts)**

* void changeDiscount(int d) - for changing discount

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  1  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0 | ===== TESTING CONSTRUCTORS ======  240312382732-WhiteSnake\_ThePurpleTour-1020  240312382732-WhiteSnake\_ThePurpleTour-1200 | ===== TESTING CONSTRUCTORS ======  240312382732-WhiteSnake\_ThePurpleTour-1020  240312382732-WhiteSnake\_ThePurpleTour-1200 |  |
|  | 2  4  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  8000  9000 | ===== TESTING METHOD vkupenPrihod() ======  0 | ===== TESTING METHOD vkupenPrihod() ======  0 |  |
|  | 2  4  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  1000  9999 | ===== TESTING METHOD vkupenPrihod() ======  3400 | ===== TESTING METHOD vkupenPrihod() ======  3400 |  |
|  | 3  6  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  1  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  1  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  0 0 0 0 0 0 0 0 1 2 3 4  APOCALYPTICA  900  0  0 0 0 0 0 0 0 0 1 2 3 4  APOCALYPTICA  900  1  10 | ===== TESTING DISCOUNT STATIC ======  240312382732-WhiteSnake\_ThePurpleTour-900  240312382732-WhiteSnake\_ThePurpleTour-1200  111112345234-Cosmosis-450  111112345234-Cosmosis-450  000000001234-APOCALYPTICA-810  000000001234-APOCALYPTICA-900 | ===== TESTING DISCOUNT STATIC ======  240312382732-WhiteSnake\_ThePurpleTour-900  240312382732-WhiteSnake\_ThePurpleTour-1200  111112345234-Cosmosis-450  111112345234-Cosmosis-450  000000001234-APOCALYPTICA-810  000000001234-APOCALYPTICA-900 |  |
|  | 2  4  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  5000  9999 | ===== TESTING METHOD vkupenPrihod() ======  1000 | ===== TESTING METHOD vkupenPrihod() ======  1000 |  |
|  | 2  4  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  2 4 0 3 1 2 3 8 2 7 3 2  WhiteSnake\_ThePurpleTour  1200  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  1 1 1 1 1 2 3 4 5 2 3 4  Cosmosis  500  0  1000  5000 | ===== TESTING METHOD vkupenPrihod() ======  2400 | ===== TESTING METHOD vkupenPrihod() ======  2400 |  |

Passed

# 26.

Implement a structure for Transaction where you store the date of the transaction (day, month and year), amount (integer) and currency (char array of 3 chars). **(10 points)**

Write a main function where you read an integer N (max 500) and then read data for N transactions (each in new line in the following format: currency amount dd/mm/yyyy). **(10 поени)**

Example:

MKD 310 18/03/2016

Print on the SO all the transactions with currency USD that have amount less then 5000 and are completed in the first 3 months of the year. **(15 поени)**

Print the total amount of these transactions. **(5 points)**

For the printing format see the output of the first test sample.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 172  USD 2701 04/11/2012  MKD 10128 08/03/2012  USD -6017 27/07/2015  MKD 9920 10/04/2013  EUR 12978 25/02/2015  USD -1711 11/05/2013  USD 2156 28/01/2014  USD 3131 19/11/2013  USD 1563 28/05/2014  USD -1609 11/12/2014  USD 7027 29/10/2013  EUR 9964 28/09/2016  EUR 7347 18/04/2013  MKD 88 01/02/2015  USD -8349 14/09/2012  MKD 2383 11/01/2013  MKD 5256 13/07/2012  EUR 9000 08/05/2014  EUR 3738 09/04/2015  USD 5965 21/02/2010  USD 13328 25/01/2014  MKD 8110 11/01/2012  MKD 1743 21/12/2010  MKD 19806 20/12/2011  USD 394 06/03/2015  USD 12124 12/01/2011  EUR 4718 07/07/2013  USD 8731 24/01/2013  USD 9072 24/10/2012  USD 8464 16/03/2013  EUR -7483 22/06/2012  EUR 3484 06/11/2016  EUR 4741 08/06/2013  EUR -3511 19/07/2016  EUR -8281 02/02/2013  EUR 18360 22/10/2013  MKD -8940 03/09/2013  EUR 8684 14/02/2015  MKD 2469 27/05/2015  MKD 11368 24/08/2015  USD -1162 12/08/2013  EUR 5870 17/12/2013  MKD 1422 05/06/2014  USD -5591 26/05/2016  USD 15696 07/01/2013  EUR 12774 18/11/2011  EUR 19006 31/08/2014  USD 3582 29/12/2015  USD 6660 01/08/2010  EUR 13010 16/09/2011  MKD -8045 03/01/2013  USD -8393 04/05/2012  USD -9975 24/03/2010  USD 2076 27/08/2013  MKD 16840 14/04/2014  USD 651 03/11/2013  MKD -5600 12/03/2014  EUR 7260 06/10/2014  EUR 8912 11/08/2010  EUR 1419 31/12/2015  MKD 834 07/12/2010  MKD 12020 06/09/2016  MKD -2847 06/09/2011  USD -7702 29/02/2011  USD 11280 24/09/2010  EUR -7303 22/04/2013  EUR 5758 23/12/2010  EUR 4135 02/08/2012  MKD 4698 19/02/2012  MKD 67 15/06/2015  MKD 14416 14/05/2013  EUR -5972 12/06/2010  USD 9715 04/06/2012  EUR 4735 08/10/2012  USD 12402 05/09/2016  USD 3901 10/06/2015  MKD -7745 19/09/2016  MKD 1400 01/03/2012  EUR 13290 31/05/2010  MKD 5458 09/01/2015  USD 3140 16/03/2013  MKD 17090 15/07/2010  EUR -2559 01/11/2012  EUR 4534 21/08/2016  MKD 9614 09/10/2016  EUR -7963 23/03/2010  MKD 925 18/08/2016  MKD -1772 09/04/2012  USD 4238 30/05/2012  MKD 17482 08/09/2011  MKD 1338 21/08/2016  MKD -8502 25/08/2014  USD 418 17/01/2016  EUR 6336 10/05/2013  MKD 5441 28/10/2012  USD -2201 31/12/2011  EUR -3986 20/09/2012  USD -6957 20/08/2014  MKD -1869 14/01/2014  USD 482 13/01/2012  USD 8452 27/02/2015  USD 10750 20/04/2013  USD -7504 23/08/2012  MKD -6861 08/05/2012  MKD 12118 24/09/2012  USD 7302 07/12/2013  USD 17810 12/02/2016  EUR -7364 13/05/2012  MKD -6527 18/01/2010  EUR 3702 06/07/2011  USD -6855 16/05/2014  USD 8337 14/05/2012  MKD 1112 25/02/2012  USD -1822 22/03/2012  EUR 3597 25/04/2012  USD -1323 08/05/2016  MKD 7530 05/04/2012  MKD 7162 22/10/2010  MKD -8774 16/05/2012  USD -7647 06/06/2016  MKD 6337 23/11/2012  USD -5546 24/06/2010  MKD 7709 06/08/2015  EUR 4810 22/10/2015  MKD -5927 12/03/2013  EUR 9722 17/08/2016  EUR 18642 06/02/2014  USD -768 01/10/2013  USD 2610 29/02/2012  USD 9574 14/06/2012  USD -671 14/09/2015  EUR 2691 15/12/2012  EUR 9198 11/02/2010  MKD 7704 06/01/2012  EUR 14894 21/12/2014  MKD -6115 28/06/2016  MKD -7922 31/02/2011  MKD -2519 30/09/2013  USD -5049 13/11/2010  MKD -886 24/06/2013  MKD 8187 23/04/2016  MKD 12350 19/10/2012  USD 18460 27/11/2016  MKD -9913 16/10/2012  EUR 8805 01/10/2014  MKD 814 27/02/2014  EUR 15994 16/09/2010  MKD 12586 07/11/2010  EUR 2958 21/10/2011  EUR 926 14/04/2015  USD 6185 29/11/2010  MKD 4732 12/03/2015  USD 15682 11/04/2016  USD -7048 25/11/2016  MKD -3805 03/03/2015  USD -5391 17/06/2014  MKD -8458 22/08/2014  USD -5913 24/07/2016  EUR 536 29/07/2013  MKD -4185 12/11/2010  MKD 18728 15/01/2013  USD 17624 19/07/2012  USD -5190 12/02/2011  MKD 5666 11/10/2012  MKD 8234 16/06/2015  MKD 713 23/11/2010  EUR 18346 27/10/2011  USD 5769 22/01/2010  EUR 13926 09/01/2012  USD 5574 27/11/2014  EUR 3824 20/08/2014  USD -7509 11/01/2013 | 1. 28/01/2014 2156 USD  2. 06/03/2015 394 USD  3. 24/03/2010 -9975 USD  4. 29/02/2011 -7702 USD  5. 16/03/2013 3140 USD  6. 17/01/2016 418 USD  7. 13/01/2012 482 USD  8. 22/03/2012 -1822 USD  9. 29/02/2012 2610 USD  10. 12/02/2011 -5190 USD  11. 11/01/2013 -7509 USD  Total: -22998 USD | 1. 28/01/2014 2156 USD  2. 06/03/2015 394 USD  3. 24/03/2010 -9975 USD  4. 29/02/2011 -7702 USD  5. 16/03/2013 3140 USD  6. 17/01/2016 418 USD  7. 13/01/2012 482 USD  8. 22/03/2012 -1822 USD  9. 29/02/2012 2610 USD  10. 12/02/2011 -5190 USD  11. 11/01/2013 -7509 USD  Total: -22998 USD |  |
|  | 315  MKD 18226 11/12/2012  EUR 19734 15/11/2014  MKD 8086 12/09/2013  USD 147 14/05/2016  MKD 19734 30/02/2014  MKD 8112 16/04/2013  EUR -6221 04/08/2013  USD 7747 05/10/2016  MKD 5943 18/06/2014  MKD 10134 01/01/2013  MKD 12580 28/12/2010  MKD 7006 11/10/2010  MKD -6016 19/04/2014  MKD -7384 30/09/2012  MKD -2461 24/07/2015  EUR 3976 09/03/2013  EUR 3194 26/02/2015  USD 8416 03/01/2016  EUR -9576 07/09/2010  MKD 9004 13/12/2016  USD 18094 09/09/2010  MKD 5324 30/01/2013  USD -713 29/04/2010  USD -2904 06/06/2015  USD 8156 22/02/2015  USD 4841 04/06/2016  USD 19670 13/11/2016  USD -4409 29/05/2014  EUR 9342 01/03/2013  USD -2833 25/05/2013  EUR 7584 20/12/2015  MKD 2718 26/06/2016  MKD 8632 26/04/2011  USD -2134 12/08/2010  USD -4847 22/04/2011  EUR 9668 18/12/2013  MKD -8243 05/08/2011  EUR -349 01/01/2016  MKD -5394 28/02/2011  MKD -5452 30/02/2010  EUR 19858 08/10/2011  MKD 2266 30/02/2014  EUR 17584 17/10/2010  MKD 12040 17/01/2013  EUR -2856 09/09/2015  USD 3112 23/09/2013  USD -9589 29/06/2010  USD 6014 03/03/2011  MKD 257 02/12/2014  MKD 9510 12/04/2015  EUR -5 03/02/2011  EUR -5350 18/05/2010  USD -1712 02/02/2011  EUR 2856 21/08/2013  USD 15524 08/03/2012  EUR 13308 02/06/2013  USD 14688 30/06/2013  MKD -1683 28/02/2015  EUR -4813 27/12/2012  MKD 7961 09/07/2013  USD 14626 21/04/2014  EUR 1701 05/12/2014  USD 4119 24/05/2011  USD -8108 28/07/2012  USD 4454 21/09/2010  MKD 11260 18/06/2012  USD 3868 10/01/2015  MKD 7930 22/09/2015  USD 11430 12/11/2011  EUR 7096 07/09/2015  EUR -6994 03/10/2014  USD -3735 17/06/2014  MKD 1290 17/09/2011  MKD 8090 17/09/2010  EUR 3895 22/06/2011  EUR -233 30/10/2012  EUR 9496 13/03/2011  USD 1976 02/09/2015  MKD -3358 25/03/2012  MKD -2078 30/05/2012  EUR 1749 21/11/2015  EUR 12046 09/12/2012  MKD 19380 26/03/2013  USD 8177 26/11/2010  EUR -6513 17/07/2012  MKD 6663 19/03/2013  EUR -596 23/03/2011  USD 13056 18/10/2011  MKD 9604 29/01/2016  EUR -9852 25/07/2015  MKD 2689 18/03/2011  EUR 15714 17/09/2010  MKD 4448 04/04/2011  USD 1883 26/10/2016  EUR 712 06/05/2011  EUR 5100 11/08/2014  MKD 614 18/11/2010  USD 3262 14/08/2014  USD 7114 15/06/2013  EUR 331 25/12/2010  EUR 7664 18/06/2015  EUR 10144 07/07/2011  USD 1276 30/11/2014  EUR 13834 18/05/2012  EUR 6473 05/11/2010  EUR 15760 18/08/2014  MKD 828 21/01/2015  EUR 9013 15/05/2010  MKD 702 19/01/2013  MKD -742 07/02/2016  MKD 15642 01/10/2010  USD 18922 21/05/2012  MKD 18854 16/09/2014  EUR -924 31/09/2013  EUR 19938 15/02/2011  EUR -3463 21/06/2014  EUR 7799 06/08/2015  USD 8962 13/04/2013  USD -3786 31/02/2010  USD 12696 04/02/2011  EUR 6753 19/10/2011  USD 3672 30/06/2012  USD -3245 31/06/2015  EUR -6367 26/12/2012  MKD 4073 03/06/2013  EUR 18666 10/06/2013  EUR 4339 23/01/2011  MKD 2574 26/11/2013  EUR 9742 30/04/2014  USD -5528 08/10/2013  USD 12738 24/07/2013  USD -579 08/07/2013  EUR 10610 17/04/2011  MKD -3465 04/06/2015  MKD -6974 30/10/2015  MKD 11924 24/04/2011  EUR 17434 14/07/2014  EUR -4708 01/08/2010  EUR -1314 29/10/2013  USD -3516 11/12/2015  MKD 3923 08/09/2011  EUR 11660 30/02/2012  USD -8057 26/02/2012  USD 5252 03/11/2014  MKD 1470 15/01/2012  USD 5775 08/08/2016  USD 3372 12/09/2013  EUR 6732 02/04/2010  USD 8190 07/03/2013  USD 6897 25/08/2014  USD 7281 26/06/2016  EUR 16456 23/01/2015  MKD 14946 25/07/2016  MKD -5981 11/01/2013  MKD 5048 25/07/2013  EUR -7149 12/03/2016  MKD 146 27/01/2011  USD 7619 18/07/2010  EUR 7670 20/08/2010  USD 60 18/09/2013  USD 4101 15/09/2012  USD 14406 12/12/2011  EUR 14314 26/06/2012  MKD 4892 27/05/2012  EUR 1674 18/06/2012  USD 2999 10/05/2013  EUR 6888 31/05/2012  EUR 8133 03/01/2014  EUR 4965 18/11/2013  MKD 1682 14/10/2011  MKD -6761 15/10/2010  EUR 13766 26/12/2011  USD -6625 04/05/2011  EUR 3169 19/06/2016  EUR -7823 11/03/2010  EUR 636 24/09/2013  USD 10830 30/10/2014  USD 462 04/08/2015  USD -5691 24/11/2014  MKD 9785 17/08/2013  MKD -1643 11/10/2013  EUR 18826 10/01/2015  MKD 8108 12/05/2011  MKD 8956 10/02/2011  EUR 7364 18/08/2010  USD -8406 23/12/2013  MKD 5164 03/12/2014  EUR 5020 08/01/2015  USD 958 09/12/2013  EUR 2388 08/06/2010  MKD -2950 12/07/2013  USD 10004 07/10/2010  EUR -6194 22/06/2016  EUR 11524 08/10/2013  MKD 2301 04/02/2012  MKD 8648 08/02/2013  MKD 15628 28/05/2013  EUR 10348 11/02/2011  MKD -5198 02/04/2013  MKD 15422 03/12/2011  USD -2418 04/03/2015  USD -9297 05/01/2010  MKD 1759 14/07/2015  MKD 3368 27/02/2016  EUR -5322 18/08/2014  USD 1074 02/05/2013  MKD -3719 29/09/2011  MKD -8006 06/11/2015  USD 4212 27/05/2012  EUR -8599 28/10/2014  USD -8542 06/04/2015  EUR -6649 21/03/2013  USD 12646 28/12/2010  USD 17370 11/09/2014  EUR 4085 01/11/2014  EUR -8544 24/06/2013  MKD 1158 10/11/2011  EUR 12000 21/11/2011  USD 1302 16/10/2012  EUR -4940 25/04/2011  MKD 8865 16/03/2014  USD 7734 27/07/2011  USD -5682 20/07/2013  MKD -7919 02/02/2016  USD 13078 14/01/2011  USD -662 29/11/2013  EUR 4818 26/08/2016  MKD 703 20/11/2010  EUR 7216 15/06/2011  EUR 18872 04/12/2014  MKD 9967 20/07/2011  EUR 3384 07/06/2014  EUR -7192 29/08/2013  MKD 15140 29/07/2015  MKD 7123 12/12/2012  USD 8446 23/05/2011  USD -7065 18/06/2015  MKD 5379 02/05/2012  EUR 13980 23/08/2016  EUR -5840 28/11/2010  MKD -5501 06/04/2013  MKD 3372 21/02/2010  EUR 18812 28/03/2013  MKD 8680 23/07/2011  EUR 12638 05/10/2012  USD 6690 25/03/2016  MKD -552 28/07/2015  MKD 5228 04/01/2010  EUR 16474 25/03/2011  MKD 9322 05/05/2013  MKD -3606 16/02/2015  USD 8938 08/02/2016  MKD 2290 13/03/2014  MKD -614 28/05/2012  USD 7408 27/08/2016  MKD 11662 14/11/2010  EUR 15272 29/09/2010  EUR 1682 14/01/2012  EUR -6814 05/03/2013  USD -6487 29/04/2011  EUR 8080 28/01/2015  USD 12274 20/12/2015  MKD 4125 31/07/2014  USD 2358 08/11/2011  USD 1561 11/01/2014  MKD -8490 09/08/2010  MKD -1067 27/03/2014  USD -7008 25/11/2014  USD 9341 31/02/2010  MKD 16282 17/11/2010  MKD 2024 13/08/2013  USD 18174 27/12/2013  USD -6416 29/10/2010  EUR 3821 22/06/2013  USD 6502 01/04/2016  USD 828 15/02/2012  USD 4694 08/07/2010  USD -3276 01/03/2010  MKD 3931 04/05/2016  MKD 3010 19/05/2013  EUR 15788 19/07/2014  MKD -6258 18/03/2015  MKD 2758 16/03/2016  MKD 2588 11/10/2014  EUR 12854 03/07/2016  USD 14030 29/12/2016  USD 4363 03/02/2015  USD 4679 12/09/2012  EUR 14966 19/01/2016  MKD 11190 30/05/2014  EUR -6950 26/09/2011  MKD 3764 10/02/2016  MKD 1196 23/11/2010  USD 7030 20/05/2015  MKD -82 25/10/2016  MKD -4942 14/05/2010  MKD 18066 27/08/2014  EUR 6124 14/10/2011  MKD -9783 20/03/2010  USD 6631 17/01/2014  EUR 1276 07/11/2010  MKD 5293 02/01/2013  USD 7776 31/07/2011  MKD -9613 21/03/2010  EUR 19488 08/02/2015  USD 4285 11/10/2011  USD 7860 06/11/2010  USD 9058 04/12/2011  USD 19256 01/11/2016  EUR 803 19/01/2016  MKD -8291 30/06/2012  MKD -3273 02/08/2013  EUR 5849 04/12/2013  MKD 14 09/03/2013  MKD 3544 29/08/2012 | 1. 02/02/2011 -1712 USD  2. 10/01/2015 3868 USD  3. 31/02/2010 -3786 USD  4. 26/02/2012 -8057 USD  5. 04/03/2015 -2418 USD  6. 05/01/2010 -9297 USD  7. 11/01/2014 1561 USD  8. 15/02/2012 828 USD  9. 01/03/2010 -3276 USD  10. 03/02/2015 4363 USD  Total: -17926 USD | 1. 02/02/2011 -1712 USD  2. 10/01/2015 3868 USD  3. 31/02/2010 -3786 USD  4. 26/02/2012 -8057 USD  5. 04/03/2015 -2418 USD  6. 05/01/2010 -9297 USD  7. 11/01/2014 1561 USD  8. 15/02/2012 828 USD  9. 01/03/2010 -3276 USD  10. 03/02/2015 4363 USD  Total: -17926 USD |  |
|  | 405  EUR 8020 05/10/2013  MKD -9692 23/01/2012  USD 2238 14/05/2015  MKD 1318 03/04/2014  USD 6705 07/05/2011  USD 1265 31/05/2015  EUR 6147 10/06/2011  USD -4722 30/04/2012  EUR 8458 19/05/2012  USD 5969 05/04/2013  EUR 1879 15/09/2012  MKD 17102 22/03/2010  USD -2708 17/02/2012  USD -680 31/02/2011  EUR 13090 04/02/2011  MKD 3010 01/11/2016  MKD -8074 03/04/2013  EUR -276 20/09/2016  EUR -8354 04/01/2016  USD 9447 05/06/2013  MKD -8352 06/08/2011  MKD -7012 05/03/2015  USD -5484 20/07/2012  USD -9579 20/11/2011  EUR 3706 23/07/2014  MKD -7632 03/07/2016  MKD -1223 16/07/2012  USD 16212 10/02/2016  USD 13052 12/09/2016  MKD 9909 29/09/2011  EUR -6599 04/11/2015  MKD 18286 27/06/2013  USD -6740 18/01/2014  USD 80 11/07/2014  EUR 7378 24/05/2013  USD 12124 27/05/2014  EUR 382 23/09/2014  EUR 7905 07/07/2011  USD 133 06/10/2014  MKD -8821 28/08/2013  MKD 11630 18/09/2015  MKD -2452 24/01/2015  MKD 2236 16/10/2013  EUR 5187 13/09/2011  USD 18260 14/11/2015  EUR -1759 13/01/2011  MKD 19398 10/05/2013  EUR 15392 18/12/2012  USD 3894 27/09/2015  USD -7704 20/11/2012  USD 14850 05/11/2016  MKD -6654 06/04/2013  MKD 15392 05/08/2015  USD 18006 29/09/2012  EUR 1900 20/10/2013  EUR 7298 22/08/2010  EUR -1815 13/12/2014  MKD 2444 25/10/2013  EUR -3241 31/02/2014  MKD 1960 27/05/2012  USD 2568 30/12/2011  MKD -6307 13/09/2013  MKD 7742 21/05/2013  EUR -378 15/03/2010  USD 8700 19/08/2016  MKD 1662 27/09/2016  MKD 2863 28/10/2014  MKD 159 03/01/2015  MKD -1811 26/01/2015  MKD 1187 06/10/2012  MKD 3536 04/03/2015  MKD 674 16/05/2012  EUR -3139 24/04/2010  EUR 7883 09/10/2015  EUR 8602 20/07/2010  MKD 4936 05/07/2010  USD -9838 18/10/2014  MKD 6957 05/06/2011  USD 17644 03/09/2015  USD -2706 31/03/2010  MKD -8334 30/02/2014  MKD -6592 30/03/2014  EUR 144 04/09/2014  USD 6196 13/03/2014  USD -6758 02/06/2013  USD 9472 19/08/2013  MKD -6057 14/11/2016  USD 5141 11/08/2010  MKD -1064 26/09/2015  USD 7160 24/11/2016  USD -1305 05/01/2015  EUR -6724 18/02/2016  EUR -2617 27/11/2012  EUR 935 08/03/2011  MKD 7462 12/10/2011  MKD 6938 27/12/2013  MKD 4252 28/08/2010  MKD 3388 27/08/2014  USD 16862 14/10/2014  MKD 9397 21/03/2014  MKD -4891 21/10/2014  EUR 948 10/07/2013  EUR 2764 03/12/2014  EUR -3475 11/02/2012  MKD -1971 13/07/2016  USD 8144 30/03/2010  USD 1708 20/06/2015  MKD -313 09/05/2016  MKD -9067 05/06/2012  USD -7716 23/03/2016  EUR 1132 05/11/2016  MKD 7696 13/09/2016  USD 4874 02/11/2012  MKD 6227 04/09/2013  MKD 16032 22/10/2014  MKD -9001 06/07/2014  MKD -7010 23/04/2011  USD 9348 13/08/2015  USD 1862 08/03/2011  MKD -8137 23/07/2011  EUR 9480 04/05/2016  USD 9134 03/01/2013  MKD 5809 30/12/2013  EUR 8529 21/02/2013  MKD 8421 06/07/2013  EUR -3450 01/09/2016  USD 3680 20/11/2010  MKD 7138 15/02/2011  EUR 18520 28/10/2010  EUR 1128 21/11/2014  MKD -6477 08/09/2014  USD 2530 27/02/2011  MKD 804 25/10/2015  EUR 14420 07/10/2013  MKD 707 07/05/2016  EUR 9979 18/11/2015  MKD 3172 28/07/2014  USD 1805 19/12/2010  USD 522 24/09/2012  USD 3460 30/11/2012  MKD 8812 30/04/2016  EUR 1760 24/05/2016  MKD -568 06/10/2010  USD 5865 02/01/2013  EUR -9422 15/01/2010  MKD 15656 27/01/2010  EUR -4399 24/01/2011  MKD 1778 11/07/2013  USD -6212 13/05/2012  EUR 4976 18/02/2015  MKD 4942 10/12/2012  EUR 4930 23/01/2011  MKD 433 30/03/2012  EUR 12276 17/08/2011  USD -3726 06/07/2012  USD -2276 19/12/2010  USD 10562 08/07/2014  MKD 9061 15/01/2012  MKD 2154 15/10/2014  EUR -9452 30/01/2014  USD 9774 29/01/2012  USD 7105 26/11/2012  USD 1941 19/09/2016  EUR 6410 29/10/2010  EUR 10536 12/09/2016  USD 2462 11/01/2013  EUR 5818 09/07/2014  USD 402 06/06/2012  USD 9088 14/08/2011  MKD 4575 02/02/2011  MKD -4177 12/05/2011  USD -7387 07/02/2010  MKD 9828 17/08/2013  EUR -3016 01/12/2012  MKD 915 17/05/2013  MKD -3252 22/05/2014  USD 9338 21/07/2010  EUR -9460 04/10/2013  EUR 5510 14/04/2011  MKD -6585 04/04/2014  MKD -4379 04/07/2010  EUR 18186 29/01/2011  EUR -3082 25/12/2010  MKD 12508 05/04/2012  USD 3302 17/06/2014  USD 14726 09/10/2016  EUR 7229 10/07/2010  MKD 157 07/12/2012  USD 4879 14/09/2013  EUR 10956 28/06/2011  USD 8746 07/03/2014  USD 3672 19/05/2010  USD -780 07/07/2014  EUR -8730 09/01/2014  EUR 3748 14/10/2010  EUR -3049 02/11/2013  E ...snip... 2015  USD 6258 26/01/2013  EUR 3228 08/01/2013  EUR 6691 21/12/2010  EUR -8442 02/11/2013  MKD 9456 15/02/2015  MKD 4073 11/08/2014  MKD 17902 20/04/2015  USD 3958 08/12/2014  MKD -7821 14/09/2014  MKD -5725 21/02/2010  MKD 5785 30/09/2015  USD -776 15/03/2011  EUR 3772 20/10/2015  USD -3871 14/03/2010  MKD 569 29/03/2013  USD 6611 14/05/2012  MKD -7347 30/02/2011  EUR 4555 07/05/2011  USD -8810 24/01/2014  EUR 1346 20/05/2010  EUR 19556 28/12/2010  USD 10586 19/08/2014  EUR 3460 03/05/2016  EUR 9484 26/03/2014  EUR 10124 14/06/2015  MKD 9677 12/05/2013  MKD 11268 13/06/2016  USD -9606 29/04/2015  MKD 3986 20/07/2012  MKD 2976 25/04/2010  EUR 16782 08/02/2011  EUR 5781 07/11/2011  USD 4677 30/04/2011  USD 5313 03/02/2016  MKD 7532 19/10/2011  EUR 887 21/02/2014  USD 547 08/08/2010  MKD -5291 30/01/2013  MKD 17386 19/07/2013  MKD 2216 25/04/2012  USD 12744 07/07/2016  EUR 19970 09/07/2013  USD -8267 21/09/2012  MKD 1347 04/10/2014  MKD -7268 19/10/2012  USD -3356 28/06/2011  EUR -8675 01/05/2015  MKD 5642 24/06/2012  MKD -6171 01/08/2012  USD 9780 25/12/2014  EUR 9806 15/09/2013  USD 9315 21/08/2012  USD 592 25/04/2010  EUR 3597 16/12/2010  EUR 3366 21/02/2016  USD 7732 04/09/2012  MKD -2998 15/09/2016  USD 8607 28/08/2011  EUR -4982 16/03/2015  MKD -1993 08/08/2010  MKD 602 22/02/2012  USD 312 12/11/2013  MKD 3164 22/12/2016  MKD 3961 05/11/2011  EUR -3015 05/02/2014  MKD 5974 06/04/2016  EUR 360 06/11/2010  EUR 3456 13/01/2013  EUR -8762 18/01/2015  EUR 12542 25/06/2015  MKD 11522 15/10/2012  EUR 2738 03/11/2011  MKD 8132 29/10/2015  MKD 19272 03/03/2014  USD 9284 20/02/2016  USD 2550 15/12/2013  USD -9509 11/09/2013  MKD -1998 09/04/2012  USD 4069 28/12/2013  MKD -694 01/11/2015  MKD -280 28/03/2011  MKD 19578 05/03/2016  MKD 6695 10/08/2011  EUR 6250 05/06/2016  MKD 18442 28/12/2011  MKD 2098 28/11/2014  USD 10350 22/07/2013  USD -8454 13/03/2015  MKD 317 18/02/2014  EUR 1098 23/05/2012  EUR 3871 07/10/2015  MKD 1229 22/07/2010  MKD 3028 02/12/2013  USD 5698 23/02/2015  MKD 9222 24/10/2010  EUR 5544 25/11/2011  USD -1557 21/07/2011  USD -4302 18/05/2016  EUR 7062 15/09/2013  MKD -7971 27/10/2013  MKD 316 15/04/2010  USD 3314 13/07/2015  EUR -303 22/12/2015  EUR 8067 06/03/2012  EUR 11954 29/11/2013  EUR 14966 02/02/2014  EUR -4194 14/11/2015  USD 3734 18/08/2015  USD 2848 18/06/2014  MKD -5125 24/01/2012  USD 2280 06/01/2012  MKD -1708 21/01/2014  MKD 4414 03/08/2014  MKD 6083 19/07/2012  EUR 9015 17/04/2015  MKD -2867 21/06/2012  EUR -3995 11/02/2010  EUR 9465 02/08/2012  MKD -9174 20/10/2016  MKD -9946 24/06/2010  EUR 2047 10/12/2012  EUR 18818 15/04/2015  USD 16250 31/07/2012  EUR 16954 28/07/2015  EUR 562 30/03/2013  EUR 6164 19/12/2011  EUR 2511 12/07/2010  USD 964 02/04/2013  EUR 1577 26/11/2016  MKD 13502 11/11/2013  USD -2604 13/12/2011  EUR 4445 27/01/2016  USD 2463 05/04/2014  USD 17774 13/12/2015  MKD 4324 14/11/2011  MKD 10916 01/09/2010  MKD -3054 06/01/2014  USD 2473 21/04/2013  EUR -9912 15/11/2013  MKD 5471 16/06/2015  USD -3175 18/11/2013  EUR 1943 21/10/2016  EUR -3705 15/04/2012  USD 6782 29/08/2014  USD 4934 20/11/2014  USD 7786 17/11/2010  USD 1738 20/11/2015  EUR 9901 03/09/2010  USD 9305 09/09/2013  MKD 7121 26/02/2013  EUR -5358 21/05/2015  USD 9217 08/06/2010  EUR -1776 21/01/2015  USD 7596 10/03/2011  MKD 14972 24/07/2010  MKD 8300 11/05/2012  USD -6893 17/03/2015  MKD 6312 31/12/2012  MKD -3759 22/01/2012  USD -9342 28/09/2016  MKD 16532 14/09/2010  USD 1550 30/12/2015  EUR 5369 09/05/2013  USD 7432 06/03/2012  EUR -4426 05/01/2012  EUR 14130 22/07/2010  USD 3391 27/12/2013  USD 44 01/11/2012  USD 7274 03/06/2012  USD -8330 09/05/2015  USD 4706 27/05/2011  USD 2675 11/03/2011  USD 6673 21/01/2012  MKD 3850 15/09/2011  EUR -5681 11/05/2014  EUR -3771 11/09/2014  EUR -5479 12/09/2014  USD 8012 02/04/2011  MKD 8336 03/09/2013  MKD -7682 17/04/2014  USD 14214 22/04/2012  EUR -5267 06/06/2012  EUR 1199 14/04/2011  EUR 16850 01/08/2016  MKD 1674 22/05/2010  USD 19878 25/04/2013  USD -8710 13/08/2013  EUR -8310 29/12/2012  EUR -9216 21/04/2013  MKD 1336 12/02/2011  MKD 1046 15/12/2010  EUR 17834 26/04/2012  EUR 3412 21/06/2013  EUR 2740 20/03/2014  EUR 13942 08/07/2015  MKD 4216 04/07/2012 | 1. 17/02/2012 -2708 USD  2. 31/02/2011 -680 USD  3. 18/01/2014 -6740 USD  4. 31/03/2010 -2706 USD  5. 05/01/2015 -1305 USD  6. 23/03/2016 -7716 USD  7. 08/03/2011 1862 USD  8. 27/02/2011 2530 USD  9. 11/01/2013 2462 USD  10. 07/02/2010 -7387 USD  11. 22/01/2016 -7555 USD  12. 06/03/2010 4684 USD  13. 30/02/2015 1507 USD  14. 15/03/2011 -776 USD  15. 14/03/2010 -3871 USD  16. 24/01/2014 -8810 USD  17. 13/03/2015 -8454 USD  18. 06/01/2012 2280 USD  19. 17/03/2015 -6893 USD  20. 11/03/2011 2675 USD  Total: -47601 USD | 1. 17/02/2012 -2708 USD  2. 31/02/2011 -680 USD  3. 18/01/2014 -6740 USD  4. 31/03/2010 -2706 USD  5. 05/01/2015 -1305 USD  6. 23/03/2016 -7716 USD  7. 08/03/2011 1862 USD  8. 27/02/2011 2530 USD  9. 11/01/2013 2462 USD  10. 07/02/2010 -7387 USD  11. 22/01/2016 -7555 USD  12. 06/03/2010 4684 USD  13. 30/02/2015 1507 USD  14. 15/03/2011 -776 USD  15. 14/03/2010 -3871 USD  16. 24/01/2014 -8810 USD  17. 13/03/2015 -8454 USD  18. 06/01/2012 2280 USD  19. 17/03/2015 -6893 USD  20. 11/03/2011 2675 USD  Total: -47601 USD |  |
|  | 104  EUR -8692 04/06/2010  USD 8882 07/02/2010  MKD -1736 18/09/2012  USD -215 24/10/2012  USD -3656 14/01/2016  USD 3345 28/06/2011  USD -8321 22/05/2015  USD 3838 19/10/2016  MKD -6067 14/02/2011  USD -1187 17/01/2012  EUR 8429 10/05/2015  USD 5708 14/10/2012  MKD 1374 15/08/2012  MKD -6358 08/11/2012  MKD 15102 15/06/2015  MKD 5224 06/06/2015  MKD 16338 05/06/2010  EUR 4742 16/01/2013  MKD 16540 08/09/2011  MKD 1472 24/09/2013  MKD 10226 26/02/2010  EUR -5352 07/05/2014  EUR 6568 19/06/2012  USD 4414 31/07/2012  EUR -4276 10/07/2012  USD -840 30/07/2011  EUR -9533 14/01/2016  MKD 3317 21/11/2014  EUR 18222 07/02/2010  USD 8620 27/03/2012  MKD -4692 13/09/2012  MKD -5027 28/11/2016  USD -5448 25/09/2014  USD 9064 07/11/2016  EUR 6176 20/11/2014  USD 2104 27/10/2011  USD 2152 08/04/2012  EUR 17958 04/04/2015  EUR -2040 20/08/2015  USD 12380 21/07/2016  USD -9987 01/10/2012  EUR -3523 06/11/2015  MKD -2145 11/06/2014  USD 6979 08/12/2010  MKD -7491 31/12/2014  MKD 4298 29/07/2014  MKD 3941 08/09/2016  EUR 6010 24/06/2014  MKD -4292 19/12/2012  MKD 5827 29/05/2014  EUR -8439 25/12/2012  MKD -9989 11/07/2014  EUR 757 16/07/2014  MKD 3506 24/08/2015  MKD -8085 08/11/2010  USD -6549 04/09/2015  EUR -6090 01/01/2011  USD 5533 20/07/2014  EUR 5364 02/08/2016  MKD 6944 12/09/2012  EUR -1572 15/08/2012  MKD 1263 24/05/2010  EUR 1162 29/07/2014  MKD -135 07/10/2012  EUR 4961 21/03/2013  MKD 1420 11/02/2010  EUR 4938 09/09/2015  EUR 3424 03/06/2012  EUR -1684 23/06/2016  EUR -4050 03/03/2014  MKD 16194 13/10/2015  USD -2589 20/09/2015  EUR 8061 20/04/2011  EUR 6241 18/06/2016  USD -2876 14/01/2010  MKD 2202 01/09/2011  MKD -6110 06/08/2014  USD 9966 05/03/2016  EUR 1858 26/12/2012  EUR 5498 03/09/2012  USD 828 25/12/2013  MKD 2827 04/12/2015  USD 8874 21/03/2014  USD 14604 07/05/2014  EUR 640 05/12/2013  EUR 8420 29/02/2010  MKD 974 02/12/2011  MKD 5924 04/07/2012  USD 9383 27/06/2015  EUR 6574 02/09/2013  EUR -9542 12/01/2011  MKD 6512 28/03/2011  USD 6034 22/07/2015  USD -8242 04/12/2014  EUR -7314 27/06/2016  EUR -3725 27/08/2013  MKD 1272 25/10/2010  USD 17494 13/10/2015  MKD 255 09/03/2015  USD 2786 17/01/2013  MKD 2208 15/04/2013  EUR 17436 18/01/2010  EUR 8634 13/06/2011  MKD 2122 21/05/2013 | 1. 14/01/2016 -3656 USD  2. 17/01/2012 -1187 USD  3. 14/01/2010 -2876 USD  4. 17/01/2013 2786 USD  Total: -4933 USD | 1. 14/01/2016 -3656 USD  2. 17/01/2012 -1187 USD  3. 14/01/2010 -2876 USD  4. 17/01/2013 2786 USD  Total: -4933 USD |  |
|  | 118  USD -403 01/12/2016  MKD 1568 26/09/2015  MKD 5932 04/07/2012  USD 4803 28/08/2013  EUR 4401 16/08/2014  USD -6404 20/05/2014  MKD 5005 06/08/2012  USD 2625 18/11/2015  EUR -4231 18/10/2015  MKD -4441 08/08/2013  MKD -3219 16/09/2014  EUR 2626 09/07/2013  EUR -4247 02/11/2014  USD 3958 07/06/2012  MKD 1117 29/03/2012  EUR -641 17/04/2012  EUR 9378 17/02/2015  MKD 3314 03/09/2015  MKD 11544 04/11/2014  EUR 13900 28/12/2014  USD 5956 11/12/2011  USD -5984 16/01/2015  USD 187 29/11/2010  USD -6706 14/07/2013  MKD 10228 04/12/2011  EUR 3330 19/10/2012  USD -6180 30/06/2012  USD 8063 13/01/2012  MKD -8630 27/09/2013  USD 1820 15/06/2012  MKD 8013 01/06/2013  EUR 9492 27/07/2016  USD 8308 10/08/2014  USD 2893 10/09/2016  USD 1107 21/02/2014  MKD 294 10/02/2016  USD -4496 08/06/2011  EUR -3126 31/10/2014  USD 4550 02/03/2011  EUR -6212 27/03/2011  EUR 4852 06/11/2010  MKD 14938 16/12/2010  USD -4343 05/01/2013  EUR 9160 02/04/2014  USD 6430 30/11/2015  MKD 18466 25/02/2016  EUR -4007 28/02/2010  EUR -7823 04/03/2012  USD 884 30/04/2015 | 1. 16/01/2015 -5984 USD  2. 21/02/2014 1107 USD  3. 02/03/2011 4550 USD  4. 05/01/2013 -4343 USD  Total: -4670 USD | 1. 16/01/2015 -5984 USD  2. 21/02/2014 1107 USD  3. 02/03/2011 4550 USD  4. 05/01/2013 -4343 USD  Total: -4670 USD |  |
|  | 386  USD 4316 24/02/2013  USD 7844 13/04/2015  EUR 2826 17/06/2010  EUR -7979 24/08/2016  MKD 7530 22/08/2016  MKD 9294 23/11/2010  MKD -9963 10/05/2012  EUR 18498 18/04/2011  EUR 771 23/10/2013  USD 19922 20/03/2012  EUR 9091 24/12/2013  USD -2953 01/07/2012  MKD 5238 21/06/2013  MKD 856 12/11/2016  USD -8455 24/03/2014  MKD 3240 27/09/2014  MKD 1951 19/07/2012  MKD 8171 14/12/2012  EUR 9761 23/11/2012  MKD 4322 23/03/2016  USD 11108 20/01/2010  MKD -9231 07/10/2010  USD -7849 26/05/2014  MKD -2783 29/04/2012  USD 1816 02/10/2013  USD -4654 09/09/2014  USD 11156 18/04/2011  MKD 14226 20/08/2016  USD 19488 21/10/2010  MKD -5642 19/07/2010  MKD 3463 28/12/2015  MKD 763 12/05/2013  EUR 8025 17/09/2010  EUR 13664 23/06/2015  EUR 906 24/02/2013  USD -165 10/07/2016  USD 8088 13/02/2013  EUR -8053 20/05/2014  MKD 18068 26/12/2016  EUR 6097 18/07/2016  EUR 1850 31/10/2011  MKD 6327 12/11/2010  EUR 2421 05/08/2010  USD -8642 08/01/2013  MKD -4555 02/05/2016  USD -2461 12/08/2014  MKD 4568 23/03/2014  USD 4879 15/12/2011  USD 8036 20/01/2012  EUR 6586 03/04/2016  EUR 6105 01/09/2015  EUR 5013 15/07/2015  MKD 13944 13/01/2013  USD -8585 07/07/2010  EUR 12190 01/01/2010  MKD 6778 16/11/2010  USD -5425 18/07/2010  EUR 19064 05/12/2016  EUR -9133 02/06/2011  MKD -1143 08/08/2015  USD 4619 10/08/2014  USD 6592 06/08/2016  USD 2368 05/03/2013  EUR 5444 14/05/2010  EUR -2249 25/05/2012  USD 4350 28/08/2013  MKD -2982 30/09/2014  EUR 6491 02/11/2012  EUR -2875 25/02/2013  EUR -5883 23/10/2012  MKD 4062 08/11/2011  USD 7728 22/04/2013  USD 5426 12/11/2015  USD 9982 16/04/2011  EUR 3274 15/10/2011  MKD -9624 27/03/2012  MKD -8604 30/12/2014  MKD 9485 15/12/2015  USD 18732 22/02/2015  USD 8980 27/01/2016  MKD -1796 01/01/2012  MKD 6894 06/02/2013  USD 2990 20/06/2014  EUR -2098 11/10/2015  USD -6514 25/07/2013  EUR 5856 27/03/2013  MKD -9375 02/04/2015  EUR -1935 03/04/2015  USD 8768 04/06/2016  EUR 314 10/02/2013  MKD 12132 28/10/2011  EUR -2992 23/10/2015  EUR -6484 10/11/2015  MKD -6172 04/06/2014  USD 14500 11/10/2010  EUR -1562 06/08/2011  USD 3271 29/02/2010  USD 6704 17/11/2010  EUR 504 04/11/2016  USD 9345 06/05/2013  EUR -993 08/06/2010  USD -9345 14/05/2011  EUR 1222 09/10/2010  MKD 9068 24/07/2015  USD -9437 28/01/2014  USD -977 10/03/2011  EUR 9490 19/07/2016  EUR -7487 10/05/2013  MKD 1521 23/08/2010  MKD -3136 05/12/2010  MKD -2152 18/04/2011  EUR -3893 11/11/2010  MKD -8123 31/06/2012  EUR 8092 03/03/2011  MKD -3623 13/06/2012  MKD 8323 30/03/2012  USD 3938 01/12/2010  EUR 14976 14/05/2012  MKD -9899 13/03/2014  EUR 15264 30/06/2016  EUR 9146 31/08/2011  MKD 5413 06/01/2014  EUR -376 10/10/2013  MKD -9933 16/08/2013  MKD -9396 17/01/2012  EUR -1252 24/01/2015  EUR 2088 26/05/2012  USD 8191 22/03/2011  USD 12868 21/12/2012  USD 1385 04/09/2012  EUR 3020 26/04/2014  USD -9416 08/04/2011  MKD 6036 20/02/2013  EUR 3089 26/04/2016  MKD 12612 28/02/2014  EUR 8109 24/03/2015  USD 6491 08/11/2013  EUR 1683 23/10/2012  EUR 17470 09/05/2015  EUR 3660 15/08/2012  EUR 780 17/06/2015  MKD -5161 09/02/2015  EUR -7840 29/07/2010  MKD 17260 26/11/2011  USD 9352 21/01/2012  MKD 3834 08/06/2011  MKD 5752 04/08/2014  USD 2348 08/02/2011  MKD -2239 21/06/2016  MKD -8028 02/07/2014  USD 10444 20/03/2013  MKD 1941 31/03/2015  EUR 3288 22/11/2014  MKD 7159 01/03/2011  MKD -1835 18/04/2011  MKD -6947 14/03/2012  MKD 9868 01/09/2011  EUR 6742 25/12/2015  EUR 2554 22/06/2011  MKD 6013 22/11/2016  EUR 1146 05/07/2013  MKD 14628 20/02/2010  USD -1231 15/03/2010  MKD -515 31/08/2010  EUR 2220 04/07/2010  USD 10798 23/07/2011  EUR 9584 13/05/2013  USD 17628 28/02/2012  MKD 8870 04/07/2011  USD 8834 25/12/2014  USD -6059 30/02/2011  USD 10000 26/11/2013  USD -4339 12/02/2011  USD 9441 22/09/2013  EUR 3170 02/04/2015  EUR 14360 24/10/2012  MKD 5641 13/01/2013  USD 12512 09/05/2010  MKD 2756 21/04/2011  USD 992 01/09/2012  MKD 13208 15/09/2015  MKD 8518 04/09/2013  USD 2158 28/02/2012  MKD 9132 06/01/2012  USD 622 17/12/2012  EUR 8406 15/02/2014  MKD 13398 23/11/2012  USD -373 07/10/2010  EUR 3340 15/04/2011  MKD 8856 13/10/2013  MKD 3221 11/08/2011  USD -9704 12/12/2016  MKD 15300 25/05/2013  MKD -9676 05/07/2010  EUR -2489 10/09/2012  USD 225 03/07/2011  USD -8740 16/12/2014  MKD 7139 08/05/2014  USD -466 11/07/2010  MKD 6781 26/08/2013  MKD 7019 05/07/2016  USD 6056 16/02/2014  USD 4394 01/06/2014  MKD 12288 12/02/2013  EUR 19760 16/01/2015  MKD 6218 11/04/2015  MKD 962 04/01/2011  EUR 19284 06/04/2014  MKD 852 10/10/2015  USD -3645 14/10/2015  USD -8732 29/05/2010  EUR 9735 22/04/2016  MKD -2004 16/06/2013  USD -8604 06/05/2011  MKD -4708 10/07/2014  EUR -764 19/08/2015  EUR 14140 07/11/2015  MKD 19166 28/02/2012  MKD -9026 05/07/2010  EUR -4909 15/05/2011  USD -9087 27/12/2015  MKD 13 24/11/2015  MKD 4135 23/08/2011  USD 8622 06/03/2010  USD -3607 21/09/2012  MKD 4043 28/01/2015  USD 780 03/11/2010  EUR 84 12/02/2010  MKD -4673 26/06/2011  EUR -6653 31/01/2011  EUR 6198 18/11/2012  MKD -5399 26/07/2014  MKD 9508 13/05/2010  MKD 16242 18/12/2016  EUR 728 26/07/2010  EUR 4234 01/12/2015  EUR -9211 19/03/2011  MKD 1420 02/06/2012  USD -9542 02/02/2010  MKD -2546 17/03/2012  EUR 15566 24/09/2015  MKD -4199 31/07/2015  USD 8296 04/02/2010  EUR -8108 11/03/2012  EUR 19770 07/02/2015  MKD -6904 15/11/2010  EUR 3582 08/05/2012  MKD 15388 14/12/2016  EUR 7264 23/01/2011  USD 1844 19/09/2015  USD 7310 20/05/2015  MKD 3810 30/12/2015  EUR 3231 03/07/2015  EUR 9103 14/01/2011  EUR -323 17/10/2013  MKD -4009 10/06/2012  USD 7156 31/08/2011  USD 147 27/04/2013  MKD -8061 10/04/2013  USD 10248 04/12/2015  MKD 2406 04/06/2015  USD 15172 29/07/2015  EUR 15314 25/07/2016  USD 10992 13/07/2013  USD -7174 11/11/2016  MKD 11914 26/04/2012  EUR -3450 28/01/2015  USD 9348 03/03/2010  MKD 9118 11/06/2014  EUR 6378 27/07/2011  MKD 1762 14/07/2015  EUR 7917 29/11/2012  EUR 6820 24/05/2016  USD 120 30/12/2013  MKD 6118 29/03/2015  MKD -3713 16/04/2012  USD -1929 22/03/2012  EUR 7711 28/03/2012  USD -4668 19/02/2016  EUR -8239 13/04/2010  EUR 3945 05/02/2010  EUR -7474 20/11/2015  EUR -8422 24/05/2016  USD -4143 23/05/2015  MKD 9324 18/04/2014  MKD 2565 02/07/2013  USD 17420 24/10/2010  MKD -7357 15/04/2014  USD -1030 02/06/2011  EUR -914 22/07/2012  USD -1986 27/03/2011  EUR -3132 22/11/2015  MKD 5167 30/02/2015  MKD 3553 18/08/2014  EUR 13670 13/12/2011  EUR 7943 21/08/2016  EUR -9048 27/08/2012  EUR 7112 29/01/2010  USD 2142 27/01/2016  USD 2318 23/10/2011  EUR 4207 31/06/2010  MKD -8911 26/10/2012  EUR 10600 27/03/2015  EUR 10972 31/08/2012  MKD 287 16/03/2012  MKD 2444 16/01/2013  EUR 11100 10/06/2012  EUR 14540 27/03/2010  MKD 17578 17/08/2015  EUR -4684 07/03/2011  MKD 9787 07/08/2011  USD 5296 08/01/2016  EUR 4880 01/01/2012  USD 4056 09/09/2010  USD 7119 18/02/2016  MKD -9864 04/03/2015  MKD 7700 07/12/2014  EUR -4163 31/12/2011  USD -2922 11/07/2014  USD 1962 09/02/2015  EUR 5318 10/01/2013  EUR 524 01/11/2012  EUR -3708 25/01/2010  EUR 1415 13/03/2015  USD 32 25/05/2016  EUR 993 15/07/2010  MKD -2767 10/04/2010  MKD -5268 27/08/2014  EUR 5301 02/07/2013  USD -9860 18/02/2011  MKD 14424 12/08/2010  EUR -9396 23/07/2010  USD 10156 08/05/2016  USD 19756 12/07/2014  MKD -5608 28/05/2011  EUR 11958 03/04/2014  MKD 303 11/07/2011  USD -9438 04/08/2015  EUR 7597 02/11/2010  USD -9516 27/05/2013  EUR -7763 09/03/2014  EUR -5918 21/07/2016  MKD 5086 11/09/2015  MKD 7428 31/01/2010  EUR 9346 22/03/2011  MKD 13968 02/12/2015  MKD -185 09/03/2013  MKD -3463 24/02/2011  MKD 7366 06/07/2012  USD 5149 09/12/2010  EUR 19356 14/01/2010  EUR 13414 15/02/2013  MKD 6 17/02/2014  MKD -2162 06/11/2015  USD 17920 02/09/2015  MKD -5448 09/04/2016  EUR 3226 08/05/2015  EUR 7525 16/01/2011  MKD 2295 07/11/2014  USD 13902 28/02/2013  EUR -3063 04/10/2014  EUR 16818 28/02/2013  MKD 3216 03/05/2014  USD 8103 23/02/2016  EUR 2090 03/06/2012  EUR 5774 07/03/2016  MKD 3603 19/06/2012  USD 14502 24/06/2016  USD 2680 14/12/2015  EUR -4758 31/08/2010  EUR -2683 25/03/2010  USD 4508 07/11/2013  USD 5647 22/11/2014  EUR 8291 02/05/2015  USD 6866 12/10/2012  USD 3258 13/01/2010  USD 326 14/01/2013  MKD -4173 04/07/2011  USD 8314 19/08/2016  MKD 10566 19/04/2010  MKD 12 09/01/2015  USD -6641 26/12/2010  EUR 18534 22/09/2016  EUR -4687 27/03/2015  MKD -1603 12/06/2015  EUR 11536 07/12/2014 | 1. 24/02/2013 4316 USD  2. 24/03/2014 -8455 USD  3. 08/01/2013 -8642 USD  4. 05/03/2013 2368 USD  5. 29/02/2010 3271 USD  6. 28/01/2014 -9437 USD  7. 10/03/2011 -977 USD  8. 08/02/2011 2348 USD  9. 15/03/2010 -1231 USD  10. 30/02/2011 -6059 USD  11. 12/02/2011 -4339 USD  12. 28/02/2012 2158 USD  13. 02/02/2010 -9542 USD  14. 22/03/2012 -1929 USD  15. 19/02/2016 -4668 USD  16. 27/03/2011 -1986 USD  17. 27/01/2016 2142 USD  18. 09/02/2015 1962 USD  19. 18/02/2011 -9860 USD  20. 13/01/2010 3258 USD  21. 14/01/2013 326 USD  Total: -44976 USD | 1. 24/02/2013 4316 USD  2. 24/03/2014 -8455 USD  3. 08/01/2013 -8642 USD  4. 05/03/2013 2368 USD  5. 29/02/2010 3271 USD  6. 28/01/2014 -9437 USD  7. 10/03/2011 -977 USD  8. 08/02/2011 2348 USD  9. 15/03/2010 -1231 USD  10. 30/02/2011 -6059 USD  11. 12/02/2011 -4339 USD  12. 28/02/2012 2158 USD  13. 02/02/2010 -9542 USD  14. 22/03/2012 -1929 USD  15. 19/02/2016 -4668 USD  16. 27/03/2011 -1986 USD  17. 27/01/2016 2142 USD  18. 09/02/2015 1962 USD  19. 18/02/2011 -9860 USD  20. 13/01/2010 3258 USD  21. 14/01/2013 326 USD  Total: -44976 USD |  |

# 27.

Implement a structure for Transaction where you store the date of the transaction (day, month and year), amount (integer) and currency (char array of 3 chars). **(10 points)**

Write a main function where you read an integer N (max 500) and then read data for N transactions (each in new line in the following format: dd/mm/yyyy amount currency). **(10 points)**

Example:

18/03/2016 310 MKD

Print on the SO all the transactions with currency MKD that have amount greater then 1000 and were completed in the period from 01.01.2013 to 31.12.2015. **(15 points)**

Print the total amount of these transactions. **(5 points)**

For the printing format see the output of the first test sample.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 346  08/03/2010 6138 USD  18/03/2014 16570 MKD  22/02/2012 1988 EUR  28/06/2015 -494 USD  18/01/2010 1639 EUR  18/04/2012 9634 MKD  06/11/2014 2777 USD  21/10/2015 10238 MKD  12/12/2010 5503 EUR  11/02/2012 -6709 USD  25/04/2016 545 MKD  04/11/2010 16884 EUR  31/04/2011 -3467 USD  29/11/2011 7776 USD  15/09/2012 5336 USD  03/12/2015 1954 USD  21/03/2012 4838 EUR  31/11/2010 -5120 USD  05/01/2011 19526 EUR  18/07/2016 -2370 MKD  25/06/2015 9185 USD  29/08/2016 10558 EUR  06/08/2010 -7048 MKD  19/11/2016 18942 EUR  19/03/2013 16022 EUR  31/01/2010 19762 USD  22/02/2014 19868 USD  20/09/2012 -9163 EUR  24/08/2011 5870 EUR  18/03/2016 -8324 USD  02/03/2014 -421 MKD  30/03/2010 -1174 USD  20/02/2011 9415 MKD  19/05/2013 -8209 MKD  23/04/2016 1974 EUR  23/06/2013 -5998 USD  07/12/2015 -5898 MKD  09/08/2016 10114 USD  08/06/2016 4850 MKD  16/11/2010 1363 USD  17/03/2015 -119 MKD  10/12/2013 3863 EUR  30/10/2014 -4865 USD  31/07/2010 7404 EUR  28/01/2014 2175 MKD  10/09/2014 5476 MKD  31/06/2016 -4843 MKD  21/01/2014 -1270 EUR  23/12/2010 8259 USD  16/08/2011 -1728 MKD  01/01/2012 6080 USD  22/08/2015 18272 EUR  30/11/2014 12166 MKD  01/08/2013 5328 MKD  20/04/2016 2243 EUR  05/07/2010 -9199 EUR  10/02/2010 16988 MKD  10/11/2010 -2362 USD  28/06/2015 -5998 USD  06/09/2012 17490 MKD  06/03/2012 7886 EUR  23/10/2010 -7023 MKD  10/07/2014 12176 MKD  25/08/2014 -9104 USD  18/08/2014 -3506 EUR  14/08/2013 -7968 USD  31/09/2014 -5934 MKD  17/02/2014 7452 MKD  16/08/2012 11616 EUR  16/05/2014 602 EUR  11/05/2015 7764 USD  02/09/2010 534 EUR  05/03/2016 -5262 EUR  15/01/2016 -6643 MKD  06/01/2012 5195 MKD  05/07/2015 6410 USD  04/07/2010 9256 USD  05/03/2014 8174 EUR  31/09/2012 3778 MKD  20/05/2014 -8828 MKD  27/07/2010 -5770 EUR  20/01/2014 -1755 EUR  08/12/2013 14804 EUR  02/02/2016 5180 USD  26/05/2013 3092 EUR  26/10/2013 -9775 MKD  19/08/2011 7722 MKD  05/03/2015 9504 EUR  15/11/2012 -2421 EUR  03/07/2012 14600 EUR  09/10/2013 17476 USD  23/02/2016 -5146 MKD  29/09/2011 14742 EUR  19/12/2016 5463 MKD  04/03/2012 -9297 MKD  28/10/2016 3896 EUR  01/03/2013 -9148 USD  10/07/2010 5071 EUR  14/07/2014 173 USD  04/05/2016 7722 USD  11/09/2013 13258 USD  20/04/2012 5890 MKD  09/07/2016 1607 MKD  05/02/2011 1194 EUR  25/01/2015 10336 MKD  18/02/2010 -9581 USD  08/08/2012 2157 USD  26/04/2010 10056 MKD  06/04/2014 8806 USD  06/08/2010 9504 EUR  29/08/2014 -6824 MKD  05/04/2011 4793 MKD  24/03/2010 6965 EUR  04/05/2014 7630 EUR  03/08/2011 2891 MKD  23/11/2015 742 EUR  19/02/2011 4550 EUR  16/02/2013 -5263 EUR  25/09/2014 5409 EUR  31/02/2016 -2757 USD  23/07/2012 3280 MKD  12/05/2011 10848 EUR  10/08/2015 -4481 MKD  21/01/2013 11142 EUR  15/03/2012 12928 EUR  26/10/2013 271 EUR  09/04/2013 9847 MKD  27/09/2011 -5898 USD  31/07/2010 -1221 EUR  27/07/2016 9560 USD  29/04/2011 5844 USD  29/03/2015 1364 MKD  27/06/2011 7418 EUR  26/05/2012 7096 EUR  12/07/2011 -3963 EUR  23/02/2010 -3316 USD  24/08/2015 2785 USD  23/11/2012 -3911 USD  24/10/2011 4100 USD  30/04/2010 9102 EUR  19/12/2013 -2590 MKD  22/06/2010 2420 MKD  18/03/2015 -8250 MKD  10/11/2011 -619 USD  16/08/2015 1702 MKD  08/03/2012 2020 MKD  19/07/2010 13330 USD  10/10/2016 -6842 MKD  16/11/2016 7097 MKD  01/04/2011 62 EUR  22/01/2012 -5898 USD  30/07/2016 -7994 USD  17/08/2012 -300 MKD  30/06/2012 -4786 MKD  01/01/2016 -4961 MKD  29/05/2014 7632 EUR  27/09/2013 9274 MKD  20/07/2016 -836 EUR  13/12/2010 -9136 MKD  25/04/2016 9470 USD  18/03/2013 2854 EUR  24/05/2013 -8648 EUR  04/11/2010 7554 MKD  16/11/2014 9552 EUR  28/08/2011 1116 EUR  06/03/2014 9152 USD  24/08/2011 -9872 USD  03/01/2011 1622 USD  23/08/2015 -4822 USD  12/08/2014 476 MKD  12/11/2013 6677 EUR  02/08/2012 15314 USD  18/09/2010 1018 USD  25/05/2011 13848 USD  30/02/2010 6664 MKD  03/09/2010 -6455 EUR  05/06/2010 -9679 MKD  14/03/2014 8881 MKD  26/11/2013 4739 MKD  29/12/2015 7302 EUR  23/04/2016 936 USD  08/04/2016 1098 USD  09/03/2016 15898 EUR  02/02/2012 8272 EUR  01/10/2011 3740 USD  07/06/2013 -9164 USD  24/09/2012 -3587 EUR  23/12/2015 -5323 EUR  31/10/2012 9020 USD  25/08/2014 8900 USD  20/12/2010 -8784 EUR  30/03/2016 6819 EUR  31/12/2014 1310 USD  29/01/2015 -632 MKD  18/09/2014 16588 EUR  08/10/2012 -8833 EUR  06/07/2012 -7346 EUR  15/09/2012 -1616 USD  13/09/2015 7535 MKD  12/04/2015 12490 MKD  15/10/2012 2863 USD  24/01/2011 9809 EUR  20/11/2010 17830 MKD  03/11/2016 16232 EUR  06/03/2016 15260 EUR  13/04/2016 -3046 EUR  10/08/2015 686 MKD  20/03/2015 -725 EUR  17/01/2010 2666 USD  11/10/2016 6600 MKD  19/07/2015 1829 EUR  08/01/2016 7928 USD  10/02/2016 -2549 EUR  29/03/2014 -6890 EUR  02/01/2013 8143 EUR  27/02/2013 10762 EUR  19/07/2013 8922 USD  13/11/2013 3840 EUR  26/06/2012 18646 USD  08/06/2010 4613 USD  08/10/2011 1714 EUR  01/02/2011 811 MKD  09/04/2015 2970 EUR  27/06/2012 2822 EUR  01/09/2010 8597 USD  15/11/2012 19506 USD  04/03/2011 -2199 EUR  29/09/2015 5803 MKD  19/06/2010 -3346 MKD  23/09/2010 -6934 USD  09/09/2012 -129 EUR  13/12/2014 6465 USD  15/08/2011 -6675 USD  07/06/2010 -2542 MKD  09/06/2016 8315 EUR  22/04/2012 5709 USD  04/12/2010 2484 EUR  07/03/2011 -3332 EUR  30/03/2015 7618 USD  26/09/2016 880 MKD  30/09/2016 5921 MKD  07/07/2015 9216 EUR  21/07/2013 3020 EUR  08/12/2015 1732 EUR  16/10/2016 5130 USD  26/11/2014 2521 MKD  08/01/2012 8208 MKD  28/11/2016 -8993 MKD  19/12/2015 -7899 MKD  23/05/2015 -5170 MKD  14/06/2010 -4 EUR  14/04/2011 -8567 USD  10/12/2011 -9762 USD  07/02/2010 5067 MKD  31/08/2014 9982 USD  04/05/2015 -2926 USD  26/04/2011 8840 EUR  26/01/2014 13388 EUR  15/03/2011 3732 EUR  17/02/2016 3026 USD  17/04/2012 11050 EUR  01/02/2011 471 MKD  19/05/2016 18938 EUR  28/08/2015 -3409 MKD  29/03/2013 1170 USD  28/11/2012 -7904 EUR  30/09/2013 6142 EUR  10/05/2014 6146 EUR  26/05/2011 850 USD  25/03/2016 -34 EUR  26/08/2011 -5861 EUR  15/11/2014 -9705 EUR  25/01/2013 1896 USD  15/07/2010 3741 EUR  18/11/2012 4072 MKD  28/01/2010 -5336 EUR  25/02/2010 -8581 USD  22/06/2014 2800 MKD  19/03/2010 -3393 EUR  14/11/2016 10174 USD  24/12/2010 -1928 USD  09/05/2016 268 MKD  29/07/2016 3631 USD  04/06/2011 17414 USD  29/07/2015 11982 USD  07/02/2014 9602 MKD  31/02/2015 -2836 EUR  03/07/2016 9288 EUR  15/05/2010 19680 EUR  20/09/2011 -7536 MKD  02/04/2014 -3085 EUR  20/12/2015 6822 EUR  18/02/2015 7957 USD  17/02/2014 12712 EUR  01/10/2011 -8781 USD  04/12/2012 19472 MKD  13/10/2014 10592 USD  30/03/2013 4469 MKD  25/09/2016 12984 MKD  25/11/2011 8191 USD  14/04/2010 15216 EUR  14/09/2011 -9993 MKD  22/07/2015 4906 MKD  07/08/2012 -4007 EUR  15/03/2014 7174 USD  11/07/2011 1721 MKD  30/03/2015 7044 USD  20/12/2015 5012 USD  25/10/2013 -7449 USD  03/12/2010 11326 MKD  17/04/2015 9452 MKD  17/05/2014 6675 USD  02/03/2011 1402 EUR  23/04/2014 15306 EUR  26/06/2015 -2411 EUR  09/11/2013 3708 MKD  08/02/2014 -9707 USD  31/05/2014 5171 EUR  09/03/2012 813 MKD  23/08/2010 18616 EUR  30/01/2016 18844 MKD  10/05/2013 16670 USD  28/11/2016 18688 EUR  17/02/2014 8572 USD  10/09/2013 14802 MKD  17/11/2016 19268 MKD  14/01/2016 13064 EUR  29/08/2011 6326 MKD  28/07/2015 6070 MKD  03/06/2010 14758 USD  04/12/2010 19954 MKD  28/10/2011 5856 USD  21/07/2016 13512 EUR  06/10/2010 3340 USD  31/01/2013 2082 EUR  24/01/2013 9852 USD  12/10/2015 -4390 USD  28/07/2014 1746 EUR  29/05/2013 7624 USD  11/07/2014 6128 EUR  05/01/2011 1658 EUR  19/12/2011 2719 USD  28/01/2016 14096 USD  01/05/2013 -1649 MKD  04/09/2012 12196 EUR  01/11/2010 -7866 USD | 1. 18/03/2014 16570 MKD  2. 21/10/2015 10238 MKD  3. 28/01/2014 2175 MKD  4. 10/09/2014 5476 MKD  5. 30/11/2014 12166 MKD  6. 01/08/2013 5328 MKD  7. 10/07/2014 12176 MKD  8. 17/02/2014 7452 MKD  9. 25/01/2015 10336 MKD  10. 09/04/2013 9847 MKD  11. 29/03/2015 1364 MKD  12. 16/08/2015 1702 MKD  13. 27/09/2013 9274 MKD  14. 14/03/2014 8881 MKD  15. 26/11/2013 4739 MKD  16. 13/09/2015 7535 MKD  17. 12/04/2015 12490 MKD  18. 29/09/2015 5803 MKD  19. 26/11/2014 2521 MKD  20. 22/06/2014 2800 MKD  21. 07/02/2014 9602 MKD  22. 30/03/2013 4469 MKD  23. 22/07/2015 4906 MKD  24. 17/04/2015 9452 MKD  25. 09/11/2013 3708 MKD  26. 10/09/2013 14802 MKD  27. 28/07/2015 6070 MKD  Total: 201882 MKD | 1. 18/03/2014 16570 MKD  2. 21/10/2015 10238 MKD  3. 28/01/2014 2175 MKD  4. 10/09/2014 5476 MKD  5. 30/11/2014 12166 MKD  6. 01/08/2013 5328 MKD  7. 10/07/2014 12176 MKD  8. 17/02/2014 7452 MKD  9. 25/01/2015 10336 MKD  10. 09/04/2013 9847 MKD  11. 29/03/2015 1364 MKD  12. 16/08/2015 1702 MKD  13. 27/09/2013 9274 MKD  14. 14/03/2014 8881 MKD  15. 26/11/2013 4739 MKD  16. 13/09/2015 7535 MKD  17. 12/04/2015 12490 MKD  18. 29/09/2015 5803 MKD  19. 26/11/2014 2521 MKD  20. 22/06/2014 2800 MKD  21. 07/02/2014 9602 MKD  22. 30/03/2013 4469 MKD  23. 22/07/2015 4906 MKD  24. 17/04/2015 9452 MKD  25. 09/11/2013 3708 MKD  26. 10/09/2013 14802 MKD  27. 28/07/2015 6070 MKD  Total: 201882 MKD |  |
|  | 210  28/07/2016 -6587 EUR  14/05/2011 1238 EUR  04/07/2011 -121 USD  24/08/2014 968 MKD  06/07/2010 -3134 EUR  19/03/2014 -1155 MKD  20/01/2013 10236 USD  14/04/2012 7148 USD  08/12/2012 -5450 MKD  26/03/2014 -1859 EUR  04/11/2014 666 USD  06/06/2016 -6291 EUR  21/03/2014 -5334 MKD  10/05/2012 11016 USD  03/12/2014 3171 USD  25/04/2015 -8508 MKD  21/09/2013 -1715 EUR  11/01/2016 8758 MKD  09/03/2016 16016 USD  16/08/2013 -5072 USD  26/06/2014 9183 USD  15/06/2010 5874 USD  09/11/2011 2258 MKD  03/01/2011 6374 USD  31/05/2016 10116 MKD  27/02/2010 17532 USD  30/02/2014 -2763 USD  08/06/2012 -6347 USD  25/02/2010 2284 MKD  19/01/2016 -8021 USD  16/11/2015 -6305 EUR  12/12/2012 13582 MKD  17/10/2014 1821 EUR  30/07/2010 8768 EUR  24/02/2013 -4293 USD  09/05/2013 1114 MKD  13/09/2012 -6859 USD  11/01/2016 1619 USD  14/07/2013 -974 EUR  08/03/2014 5183 USD  24/08/2013 5200 USD  18/10/2013 -2150 MKD  16/06/2016 -3740 EUR  07/02/2010 -2601 MKD  27/12/2014 1972 EUR  06/10/2011 5740 EUR  19/01/2016 7128 USD  08/08/2010 6660 USD  22/05/2015 5752 MKD  14/01/2010 5670 EUR  27/03/2012 -1897 EUR  12/12/2015 2704 MKD  16/12/2015 -7152 EUR  03/07/2011 17126 MKD  16/07/2010 16678 MKD  18/06/2010 -5818 USD  24/01/2011 8592 USD  26/03/2010 12116 USD  19/09/2016 8460 MKD  04/07/2013 -3249 USD  16/10/2013 4876 MKD  06/06/2013 4436 MKD  03/03/2014 4842 USD  22/02/2013 7466 EUR  01/10/2014 -8257 MKD  05/02/2012 4092 EUR  06/09/2010 4504 USD  28/01/2012 13208 MKD  31/03/2012 2445 EUR  26/12/2015 13496 EUR  14/09/2016 -8970 USD  22/03/2013 -1871 EUR  29/03/2010 1244 EUR  30/07/2012 7564 USD  03/05/2013 -7383 MKD  19/07/2014 312 USD  10/02/2010 9977 USD  06/08/2012 300 USD  16/11/2012 15388 USD  07/02/2014 -5150 MKD  16/01/2010 5600 EUR  02/05/2011 8442 EUR  12/09/2011 7189 EUR  29/08/2016 150 USD  13/02/2010 -2404 USD  08/10/2011 6242 EUR  09/02/2012 7066 EUR  10/04/2011 -4965 USD  25/04/2013 4696 EUR  13/05/2011 9423 EUR  27/04/2015 13356 USD  22/11/2013 19216 USD  07/03/2012 -6917 USD  09/10/2010 -5202 EUR  02/02/2016 7588 USD  24/08/2016 7400 EUR  10/11/2013 1540 USD  17/11/2016 2801 USD  22/10/2010 7440 EUR  12/05/2011 1002 USD  28/03/2010 4177 USD  28/06/2015 -2843 USD  11/08/2014 18926 USD  19/10/2010 7987 EUR  17/11/2010 -4990 EUR  21/12/2012 -8839 EUR  17/02/2016 -6943 MKD  21/06/2012 16330 USD  06/06/2011 4155 USD  24/01/2014 816 MKD  13/08/2013 -1225 MKD  29/10/2016 4602 EUR  11/05/2013 7788 MKD  11/02/2010 3833 EUR  25/07/2013 2078 MKD  12/08/2014 825 MKD  17/12/2016 -5437 EUR  02/11/2013 3692 USD  06/03/2012 8842 EUR  06/12/2015 6698 MKD  03/06/2010 -4726 EUR  30/06/2015 14476 MKD  14/04/2013 -5191 USD  24/09/2016 8811 USD  19/01/2010 -1134 MKD  03/02/2015 8725 USD  21/07/2014 -4730 USD  04/04/2013 -2973 EUR  01/02/2016 -9373 EUR  16/04/2014 4930 USD  09/10/2016 -9622 MKD  14/03/2015 7522 MKD  04/04/2011 11764 EUR  13/09/2011 3376 EUR  30/04/2010 9302 MKD  08/12/2014 6073 USD  16/06/2015 9298 EUR  15/10/2010 10260 EUR  03/10/2011 -4129 USD  31/10/2012 5716 MKD  24/04/2012 4684 USD  20/11/2016 -7086 EUR  09/02/2014 14086 MKD  22/11/2016 16472 MKD  20/07/2015 17256 USD  09/01/2010 9348 EUR  26/01/2016 18514 EUR  02/04/2015 -4368 USD  05/06/2013 4634 EUR  10/11/2012 4408 MKD  30/06/2016 12508 EUR  10/04/2014 -5745 MKD  27/11/2010 14514 MKD  09/05/2014 15054 EUR  19/02/2016 8260 USD  07/05/2014 1632 MKD  19/07/2014 5527 MKD  21/04/2015 2435 USD  22/08/2011 797 USD  30/10/2011 9639 MKD  03/02/2011 -6774 MKD  29/12/2015 181 USD  28/08/2016 7048 MKD  06/02/2010 12004 EUR  05/09/2010 -3748 EUR  17/04/2010 18202 MKD  11/12/2016 5800 MKD  24/08/2015 2957 MKD  27/05/2015 -1930 MKD  05/02/2012 788 EUR  24/08/2011 12510 USD  30/07/2014 8712 USD  26/06/2013 7715 USD  02/01/2010 18744 EUR  31/02/2013 1808 MKD  16/02/2013 16254 USD  31/07/2016 11564 USD  21/05/2015 -1112 EUR  29/07/2014 15932 EUR  22/02/2014 19686 MKD  22/12/2011 4938 EUR  03/10/2011 8803 MKD  24/05/2016 12896 MKD  31/02/2015 -7177 USD  27/09/2013 16484 USD  24/06/2012 -1831 USD  22/08/2016 10742 USD  10/01/2011 4435 MKD  13/07/2013 -2549 MKD  05/09/2014 12216 USD  27/03/2011 9779 MKD  29/07/2013 -5497 EUR  21/03/2013 2954 MKD  12/04/2016 9178 MKD  28/09/2015 1418 EUR  01/11/2013 -1379 USD  19/05/2011 3878 MKD  18/07/2013 -5014 USD  13/01/2014 -7027 MKD  28/07/2012 -8578 USD  21/10/2016 3078 EUR  08/08/2014 3928 MKD  11/03/2014 -632 EUR  14/12/2011 -8626 USD  17/02/2011 5958 USD  23/08/2016 10626 EUR  12/04/2010 2980 EUR  13/12/2010 3772 MKD  17/03/2014 5412 EUR  26/03/2015 28 EUR | 1. 09/05/2013 1114 MKD  2. 22/05/2015 5752 MKD  3. 12/12/2015 2704 MKD  4. 16/10/2013 4876 MKD  5. 06/06/2013 4436 MKD  6. 11/05/2013 7788 MKD  7. 25/07/2013 2078 MKD  8. 06/12/2015 6698 MKD  9. 30/06/2015 14476 MKD  10. 14/03/2015 7522 MKD  11. 09/02/2014 14086 MKD  12. 07/05/2014 1632 MKD  13. 19/07/2014 5527 MKD  14. 24/08/2015 2957 MKD  15. 31/02/2013 1808 MKD  16. 22/02/2014 19686 MKD  17. 21/03/2013 2954 MKD  18. 08/08/2014 3928 MKD  Total: 110022 MKD | 1. 09/05/2013 1114 MKD  2. 22/05/2015 5752 MKD  3. 12/12/2015 2704 MKD  4. 16/10/2013 4876 MKD  5. 06/06/2013 4436 MKD  6. 11/05/2013 7788 MKD  7. 25/07/2013 2078 MKD  8. 06/12/2015 6698 MKD  9. 30/06/2015 14476 MKD  10. 14/03/2015 7522 MKD  11. 09/02/2014 14086 MKD  12. 07/05/2014 1632 MKD  13. 19/07/2014 5527 MKD  14. 24/08/2015 2957 MKD  15. 31/02/2013 1808 MKD  16. 22/02/2014 19686 MKD  17. 21/03/2013 2954 MKD  18. 08/08/2014 3928 MKD  Total: 110022 MKD |  |
|  | 483  17/10/2011 8176 EUR  31/02/2010 190 EUR  20/02/2011 -2001 USD  10/11/2015 8512 EUR  05/02/2012 12954 USD  19/07/2013 8236 EUR  19/10/2010 6560 MKD  10/02/2015 9582 EUR  18/01/2016 -3954 USD  14/09/2015 6784 USD  11/05/2013 4620 EUR  18/03/2015 5404 MKD  03/06/2013 9622 USD  09/05/2015 2772 EUR  05/05/2012 712 MKD  29/07/2016 -5301 USD  17/02/2011 18396 MKD  16/01/2011 13480 EUR  25/03/2012 1984 MKD  11/10/2015 18192 EUR  14/04/2013 4747 EUR  05/01/2016 -7545 USD  16/03/2014 5783 USD  17/10/2010 9589 EUR  23/11/2011 450 MKD  05/07/2015 1852 USD  10/11/2011 6474 USD  12/05/2015 -5647 EUR  09/04/2016 -8186 MKD  23/11/2015 7750 USD  07/12/2013 7272 USD  23/06/2013 -4938 USD  07/09/2013 -7986 USD  12/12/2016 8185 EUR  08/03/2013 7914 EUR  11/06/2014 3426 EUR  30/07/2010 6447 EUR  29/08/2013 15036 MKD  31/12/2012 -2761 MKD  18/01/2013 9891 MKD  19/03/2011 -3977 USD  27/01/2011 -7489 EUR  23/08/2015 4848 EUR  06/09/2015 5754 MKD  01/04/2010 18266 USD  01/06/2011 -7549 MKD  18/06/2014 15684 USD  17/02/2016 -7212 MKD  20/12/2012 7224 EUR  15/03/2015 10864 USD  13/04/2012 1764 USD  17/10/2011 18358 EUR  26/04/2015 -9492 MKD  21/12/2013 9472 EUR  26/08/2012 -4231 USD  22/02/2015 2800 EUR  16/03/2014 852 EUR  16/01/2016 -3495 EUR  22/12/2016 -2177 EUR  10/12/2011 485 MKD  31/07/2015 10472 USD  31/09/2015 15194 USD  07/06/2015 -9485 MKD  09/09/2010 -3318 USD  10/02/2011 1744 EUR  08/06/2015 2208 EUR  01/07/2015 11858 MKD  01/04/2016 5691 MKD  16/01/2012 8611 MKD  26/09/2011 3222 EUR  10/02/2011 574 EUR  20/05/2015 8839 USD  04/02/2013 -4749 EUR  01/12/2013 9822 MKD  09/10/2014 -481 USD  24/01/2011 7024 USD  12/12/2011 5604 USD  31/02/2013 8507 EUR  22/01/2011 -2652 EUR  04/01/2014 -4463 EUR  28/05/2013 684 MKD  20/07/2016 -7051 EUR  29/07/2015 -164 MKD  16/05/2013 818 MKD  01/05/2016 4216 USD  12/05/2016 2824 MKD  19/01/2014 -5170 USD  15/08/2013 4280 MKD  29/06/2016 8243 USD  29/09/2011 895 USD  17/03/2012 1692 USD  27/09/2011 9647 USD  09/04/2015 8929 MKD  23/02/2016 4976 EUR  13/10/2015 8318 USD  25/10/2012 12564 USD  16/02/2016 -7380 USD  24/11/2014 -1532 EUR  14/09/2015 -3090 EUR  10/11/2011 10160 EUR  07/05/2012 14090 USD  20/12/2012 -7354 EUR  06/02/2010 1483 MKD  06/05/2012 -3862 MKD  19/03/2014 5622 MKD  12/08/2012 1446 MKD  24/08/2015 5425 EUR  01/08/2015 -7641 MKD  20/10/2014 373 USD  20/06/2014 -7527 MKD  06/11/2014 1668 MKD  04/05/2016 3685 USD  31/12/2014 -9956 EUR  29/07/2013 871 USD  11/06/2015 2610 MKD  27/05/2012 3256 MKD  19/11/2015 -4172 USD  31/08/2014 9686 MKD  20/01/2016 8760 USD  17/09/2015 -5281 USD  14/03/2016 8763 USD  25/01/2016 5405 MKD  05/01/2016 15106 EUR  02/03/2013 7170 USD  16/02/2016 16596 USD  28/01/2015 7890 MKD  12/05/2014 1506 EUR  11/03/2011 -295 MKD  20/03/2012 4881 MKD  06/07/2013 6614 USD  13/11/2016 5999 USD  05/02/2010 4622 MKD  30/04/2010 848 EUR  19/05/2011 6299 EUR  14/07/2014 15758 USD  17/09/2014 -6358 EUR  23/05/2011 6469 EUR  10/11/2010 -8884 EUR  08/11/2016 1696 EUR  08/06/2010 618 EUR  01/01/2010 3709 MKD  28/10/2015 14252 USD  14/10/2014 -4501 MKD  26/09/2010 9586 USD  26/03/2013 2244 USD  25/06/2011 4186 EUR  16/07/2016 17696 MKD  18/09/2010 -7417 MKD  18/12/2012 15790 MKD  06/05/2016 3874 EUR  03/05/2012 3427 MKD  23/06/2013 -7779 USD  19/02/2012 8264 USD  14/06/2011 18826 MKD  23/03/2013 -1211 MKD  29/04/2015 2703 EUR  05/05/2011 738 EUR  11/04/2013 -9681 MKD  03/01/2010 8365 USD  29/05/2016 -4938 MKD  08/05/2010 -7620 EUR  13/08/2010 8587 MKD  08/06/2011 -5896 USD  09/09/2012 8133 MKD  27/03/2011 18658 MKD  27/08/2014 -7908 MKD  31/06/2013 -225 USD  06/02/2010 -7549 EUR  14/12/2015 -2109 MKD  21/11/2014 -8648 MKD  03/06/2013 -8375 USD  13/08/2012 17614 EUR  29/01/2010 -8180 USD  21/05/2010 -7895 MKD  04/02/2016 -1637 USD  06/12/2014 10214 EUR  30/07/2016 -430 EUR  02/04/2010 -5196 MKD  09/08/2013 -1360 USD  26/09/2011 1916 MKD  17/04/2011 -7928 EUR  14/01/2012 8899 MKD  21/03/2012 -5157 MKD  15/08/2011 -9752 EUR  02/01/2014 4358 MKD  18/07/2014 519 EUR  01/04/2012 7656 USD  26/02/2013 19140 MKD  15/08/2016 -4933 USD  05/08/2016 916 USD  25/09/2012 17586 EUR  26/01/2012 9504 EUR  30/02/2010 -7477 MKD  20/12/2013 242 EUR  10/08/2012 5567 EUR  16/07/2011 4357 USD  2 ...snip... 2 -3777 USD  27/12/2010 -565 EUR  27/12/2012 -3070 EUR  31/11/2016 15792 EUR  23/06/2011 9560 EUR  25/04/2014 13896 USD  10/09/2011 -3518 MKD  20/08/2015 15622 EUR  09/08/2014 4818 MKD  04/03/2016 5814 USD  08/02/2013 9316 MKD  25/07/2011 6894 USD  29/04/2010 -3100 USD  26/08/2014 15878 MKD  11/09/2012 -5836 MKD  14/02/2010 -5022 EUR  03/04/2011 10652 MKD  25/08/2014 -4206 USD  24/09/2016 9508 EUR  28/08/2016 15582 EUR  16/07/2012 -3913 USD  05/10/2016 2770 EUR  14/11/2014 15294 EUR  29/01/2010 6197 USD  10/05/2013 -7045 MKD  01/10/2010 14286 EUR  30/10/2016 -7055 USD  25/05/2015 9056 MKD  19/03/2012 6654 USD  12/08/2016 -3553 EUR  13/10/2010 -6198 USD  11/10/2011 7813 EUR  04/01/2013 4966 USD  04/01/2014 3376 MKD  07/10/2011 4582 USD  27/09/2014 -3617 EUR  01/03/2015 -4204 MKD  25/07/2013 9720 USD  21/02/2015 9884 EUR  22/02/2015 -1157 EUR  15/09/2016 -8954 EUR  26/07/2012 9967 EUR  06/01/2016 -6156 EUR  25/08/2016 6188 USD  02/08/2010 -3994 MKD  11/08/2014 8275 MKD  23/04/2012 6360 USD  04/11/2013 7426 EUR  04/01/2010 4464 MKD  17/04/2014 -7555 USD  01/03/2015 6617 EUR  03/05/2011 -289 EUR  13/03/2010 13782 USD  11/10/2014 -3163 MKD  15/03/2012 8580 EUR  11/07/2012 9887 EUR  19/04/2014 -8764 EUR  17/04/2016 4170 MKD  15/02/2016 -5017 MKD  18/06/2010 3966 MKD  24/08/2012 2816 USD  31/04/2014 1025 USD  09/02/2015 6233 MKD  17/04/2015 -2177 USD  04/03/2013 3228 EUR  04/07/2016 -3621 MKD  14/05/2012 14208 EUR  23/10/2014 2822 EUR  02/01/2014 -6035 MKD  10/11/2011 819 USD  27/05/2016 2476 MKD  23/06/2011 18418 EUR  11/07/2011 17152 MKD  30/10/2012 1519 EUR  25/07/2015 6449 EUR  07/03/2010 -8067 USD  02/10/2010 8148 EUR  30/10/2015 6703 USD  02/07/2014 7226 USD  04/10/2012 -4932 EUR  09/01/2016 3755 USD  08/11/2012 9536 EUR  03/04/2012 -8307 MKD  03/02/2014 -8753 MKD  06/09/2013 485 EUR  01/06/2014 814 USD  27/01/2011 16080 EUR  14/07/2015 -6765 MKD  23/09/2013 -3755 MKD  19/06/2016 -1919 MKD  17/03/2011 -9735 USD  24/08/2012 2370 MKD  12/02/2014 3186 EUR  24/09/2014 3377 MKD  12/06/2015 14688 USD  16/01/2014 -147 EUR  30/03/2015 2801 USD  11/09/2016 5538 EUR  01/01/2013 3424 USD  14/03/2010 1730 MKD  11/12/2015 -8343 EUR  17/09/2014 9516 USD  31/01/2010 18 MKD  01/01/2010 3924 USD  09/04/2012 -1493 EUR  26/06/2011 5766 USD  27/11/2010 -147 MKD  29/06/2015 18552 MKD  10/01/2014 9372 USD  08/04/2013 17284 USD  23/05/2015 352 MKD  24/04/2012 8948 EUR  30/06/2012 -3700 USD  22/09/2016 7240 EUR  06/05/2014 1560 EUR  17/12/2015 -9648 MKD  15/03/2013 12148 USD  14/08/2013 19394 USD  14/12/2016 -3087 USD  29/04/2013 -3292 MKD  04/11/2013 3338 MKD  02/12/2012 -752 MKD  10/08/2014 -9961 USD  24/04/2013 3556 MKD  26/08/2015 1502 EUR  05/11/2011 -5116 EUR  13/05/2013 -579 USD  30/12/2012 -2784 MKD  06/02/2011 8008 EUR  31/09/2016 495 USD  09/09/2011 5297 EUR  01/07/2010 -1478 EUR  10/03/2014 -9655 MKD  03/01/2014 -8937 MKD  04/06/2012 -8110 MKD  26/08/2010 9550 MKD  15/04/2014 1022 USD  11/03/2011 2574 MKD  21/10/2015 -3652 MKD  22/09/2016 8896 EUR  28/11/2010 16684 MKD  10/03/2014 7118 MKD  30/03/2013 13382 USD  29/07/2016 -4267 MKD  14/01/2013 -2008 USD  20/12/2011 10028 USD  08/04/2016 -7812 USD  23/04/2016 -5777 USD  11/03/2011 18682 USD  03/04/2016 1395 USD  10/02/2011 -4659 USD  23/11/2013 2381 EUR  12/03/2016 10386 USD  07/05/2016 6566 MKD  18/04/2013 -9909 EUR  07/05/2015 12382 USD  02/08/2014 -2851 MKD  06/08/2012 -6970 EUR  30/03/2014 -2646 MKD  31/05/2013 1799 MKD  05/04/2013 -6041 MKD  11/10/2013 793 EUR  29/06/2014 9906 EUR  13/03/2015 4222 MKD  06/01/2013 -2207 USD  09/01/2015 8423 EUR  30/07/2011 -4188 MKD  23/04/2011 16078 USD  27/09/2015 -5066 EUR  26/01/2016 6740 EUR  09/07/2012 1100 EUR  06/10/2016 -7646 EUR  14/11/2011 3992 USD  31/03/2012 -227 EUR  03/10/2016 -4225 MKD  19/09/2011 19078 USD  15/04/2013 9896 MKD  08/06/2015 10382 USD  10/05/2012 1814 EUR  21/10/2014 7351 USD  09/08/2011 9194 EUR  24/01/2016 -2356 MKD  23/06/2010 -764 MKD  21/12/2012 18262 EUR  25/03/2014 2903 USD  28/12/2011 6400 MKD  16/01/2016 5997 USD  10/09/2013 1995 USD  02/12/2014 -145 EUR  22/02/2016 -53 MKD  08/09/2015 804 EUR  25/11/2012 -1422 MKD  20/04/2016 -9890 EUR  27/02/2015 3375 MKD  20/01/2014 108 USD  03/04/2015 16366 USD | 1. 18/03/2015 5404 MKD  2. 29/08/2013 15036 MKD  3. 18/01/2013 9891 MKD  4. 06/09/2015 5754 MKD  5. 01/07/2015 11858 MKD  6. 01/12/2013 9822 MKD  7. 15/08/2013 4280 MKD  8. 09/04/2015 8929 MKD  9. 19/03/2014 5622 MKD  10. 06/11/2014 1668 MKD  11. 11/06/2015 2610 MKD  12. 31/08/2014 9686 MKD  13. 28/01/2015 7890 MKD  14. 02/01/2014 4358 MKD  15. 26/02/2013 19140 MKD  16. 27/05/2014 8386 MKD  17. 03/06/2015 3664 MKD  18. 19/08/2015 16066 MKD  19. 08/12/2013 8048 MKD  20. 17/12/2015 7392 MKD  21. 20/11/2013 6578 MKD  22. 29/11/2013 8926 MKD  23. 06/04/2014 17714 MKD  24. 27/06/2015 6043 MKD  25. 09/08/2014 4818 MKD  26. 08/02/2013 9316 MKD  27. 26/08/2014 15878 MKD  28. 25/05/2015 9056 MKD  29. 04/01/2014 3376 MKD  30. 11/08/2014 8275 MKD  31. 09/02/2015 6233 MKD  32. 24/09/2014 3377 MKD  33. 29/06/2015 18552 MKD  34. 04/11/2013 3338 MKD  35. 24/04/2013 3556 MKD  36. 10/03/2014 7118 MKD  37. 31/05/2013 1799 MKD  38. 13/03/2015 4222 MKD  39. 15/04/2013 9896 MKD  40. 27/02/2015 3375 MKD  Total: 316950 MKD | 1. 18/03/2015 5404 MKD  2. 29/08/2013 15036 MKD  3. 18/01/2013 9891 MKD  4. 06/09/2015 5754 MKD  5. 01/07/2015 11858 MKD  6. 01/12/2013 9822 MKD  7. 15/08/2013 4280 MKD  8. 09/04/2015 8929 MKD  9. 19/03/2014 5622 MKD  10. 06/11/2014 1668 MKD  11. 11/06/2015 2610 MKD  12. 31/08/2014 9686 MKD  13. 28/01/2015 7890 MKD  14. 02/01/2014 4358 MKD  15. 26/02/2013 19140 MKD  16. 27/05/2014 8386 MKD  17. 03/06/2015 3664 MKD  18. 19/08/2015 16066 MKD  19. 08/12/2013 8048 MKD  20. 17/12/2015 7392 MKD  21. 20/11/2013 6578 MKD  22. 29/11/2013 8926 MKD  23. 06/04/2014 17714 MKD  24. 27/06/2015 6043 MKD  25. 09/08/2014 4818 MKD  26. 08/02/2013 9316 MKD  27. 26/08/2014 15878 MKD  28. 25/05/2015 9056 MKD  29. 04/01/2014 3376 MKD  30. 11/08/2014 8275 MKD  31. 09/02/2015 6233 MKD  32. 24/09/2014 3377 MKD  33. 29/06/2015 18552 MKD  34. 04/11/2013 3338 MKD  35. 24/04/2013 3556 MKD  36. 10/03/2014 7118 MKD  37. 31/05/2013 1799 MKD  38. 13/03/2015 4222 MKD  39. 15/04/2013 9896 MKD  40. 27/02/2015 3375 MKD  Total: 316950 MKD |  |
|  | 50  08/02/2014 6759 MKD  04/06/2010 -883 MKD  16/01/2011 16664 MKD  03/12/2014 6156 MKD  27/01/2016 -7364 USD  19/07/2010 78 MKD  27/03/2013 -9708 EUR  14/07/2013 9784 MKD  05/11/2016 9228 EUR  15/11/2015 -1260 USD  08/11/2010 -968 USD  08/11/2015 18994 USD  01/08/2015 3381 MKD  08/04/2011 849 EUR  20/04/2013 -477 USD  10/11/2012 -7456 USD  19/09/2016 2796 USD  24/08/2013 -262 USD  11/06/2013 1190 USD  27/07/2015 7412 EUR  05/08/2010 -9639 MKD  17/09/2016 -8304 EUR  29/05/2015 9580 EUR  13/08/2010 8411 MKD  16/08/2011 -6812 USD  14/02/2010 13170 EUR  05/08/2015 3186 EUR  24/10/2016 -9520 EUR  28/07/2011 2742 EUR  02/08/2011 -553 USD  13/11/2010 3873 MKD  09/10/2010 3074 EUR  27/02/2010 18584 USD  16/12/2015 -506 USD  18/11/2015 2558 MKD  24/11/2013 -3251 USD  16/09/2013 -3630 EUR  05/07/2011 14022 MKD  19/06/2013 -5901 USD  01/03/2015 -167 EUR  16/07/2013 -7333 USD  17/02/2012 8683 USD  16/01/2011 7883 EUR  16/07/2011 -6324 EUR  17/04/2010 -9484 MKD  30/05/2011 -8786 USD  22/02/2014 229 EUR  22/07/2014 -4614 USD  21/03/2011 272 MKD  14/05/2012 -2955 USD | 1. 08/02/2014 6759 MKD  2. 03/12/2014 6156 MKD  3. 14/07/2013 9784 MKD  4. 01/08/2015 3381 MKD  5. 18/11/2015 2558 MKD  Total: 28638 MKD | 1. 08/02/2014 6759 MKD  2. 03/12/2014 6156 MKD  3. 14/07/2013 9784 MKD  4. 01/08/2015 3381 MKD  5. 18/11/2015 2558 MKD  Total: 28638 MKD |  |
|  | 312  06/06/2014 9968 USD  08/05/2012 15674 EUR  16/07/2012 7170 USD  27/07/2011 7082 EUR  27/05/2013 -8242 USD  07/05/2015 9609 USD  27/08/2014 5024 USD  27/10/2016 8027 USD  13/11/2013 5921 USD  25/06/2015 -9806 EUR  17/06/2013 4227 EUR  13/09/2014 7913 USD  09/05/2015 6899 EUR  18/10/2016 967 MKD  25/12/2012 -1181 EUR  27/06/2014 360 USD  08/11/2011 13494 USD  06/02/2014 -9356 USD  14/02/2013 -9026 EUR  05/01/2013 -678 EUR  11/02/2010 386 USD  20/09/2014 1283 EUR  23/06/2015 6013 MKD  08/10/2011 135 USD  16/10/2011 15454 EUR  24/05/2011 4583 MKD  11/11/2014 -9079 USD  19/08/2012 316 USD  10/12/2014 10780 MKD  19/08/2016 6702 MKD  03/11/2011 3906 EUR  10/03/2010 -8631 MKD  07/04/2014 7374 USD  16/01/2012 -1499 EUR  25/08/2010 -3376 EUR  20/03/2016 7146 USD  31/05/2015 4597 EUR  08/12/2014 11744 USD  30/08/2013 -411 USD  29/01/2010 3236 MKD  14/10/2013 280 MKD  10/05/2011 13762 USD  05/10/2014 -6858 EUR  27/01/2015 -3422 MKD  09/06/2013 11948 USD  04/01/2016 3072 MKD  22/07/2013 -9847 USD  10/08/2014 -1877 MKD  16/02/2015 9323 MKD  20/09/2015 -7054 EUR  02/02/2016 13528 MKD  28/06/2014 1419 MKD  07/03/2014 7187 MKD  07/08/2012 12798 EUR  17/03/2015 5552 USD  16/10/2015 4233 MKD  21/03/2014 4326 MKD  26/08/2016 13986 EUR  07/07/2010 17656 USD  26/07/2013 4034 USD  02/04/2010 46 EUR  29/08/2011 -765 MKD  20/12/2016 14490 MKD  28/03/2015 -4159 MKD  30/10/2015 10560 EUR  14/01/2010 3607 USD  31/10/2014 97 EUR  17/01/2016 -5487 EUR  02/09/2015 3983 USD  24/06/2015 -8138 MKD  30/01/2016 8272 USD  07/09/2011 -7880 MKD  23/10/2014 14844 EUR  11/11/2015 -7392 MKD  17/02/2016 335 USD  25/04/2011 2606 USD  09/09/2015 -3461 MKD  20/02/2012 1595 EUR  16/02/2011 8606 MKD  06/08/2014 -2657 USD  31/09/2014 -122 MKD  01/12/2011 8272 USD  24/10/2012 2538 EUR  16/08/2012 17388 MKD  09/05/2015 15114 USD  03/07/2016 4779 EUR  23/11/2010 13118 MKD  07/02/2015 2261 EUR  08/01/2010 -5129 MKD  26/02/2013 13112 USD  14/11/2011 -9123 MKD  28/06/2010 7370 USD  05/07/2010 5394 EUR  05/11/2014 6957 EUR  05/02/2013 9422 EUR  21/12/2012 4730 EUR  25/12/2010 7721 USD  12/03/2011 4236 USD  05/06/2012 -6243 USD  09/02/2011 16486 USD  25/03/2012 104 MKD  15/07/2016 17846 EUR  09/09/2015 16252 EUR  21/11/2013 -7997 MKD  10/12/2016 -7866 MKD  12/11/2013 17082 MKD  03/09/2013 6710 MKD  07/06/2013 -7951 EUR  27/07/2015 -2063 MKD  30/06/2014 7896 MKD  22/02/2015 8211 MKD  30/03/2014 -7010 USD  01/02/2012 -4846 EUR  19/01/2013 8955 USD  31/01/2012 11818 EUR  17/07/2015 -8037 USD  15/05/2013 5652 MKD  10/11/2014 4905 EUR  14/10/2014 8272 MKD  01/11/2010 528 USD  15/09/2015 2780 EUR  13/12/2011 19164 EUR  17/11/2010 2141 EUR  25/04/2011 6388 MKD  30/08/2016 -2564 USD  23/03/2016 18242 USD  29/12/2014 -9262 MKD  22/03/2015 4886 MKD  07/07/2013 -1484 MKD  31/09/2012 8385 MKD  24/07/2016 17348 EUR  23/03/2012 8230 USD  01/10/2013 8180 EUR  04/12/2011 -7643 EUR  22/03/2014 -138 USD  19/08/2016 7283 MKD  04/07/2013 5383 MKD  30/04/2012 2928 EUR  06/02/2014 6221 MKD  30/09/2010 -3610 USD  29/04/2013 6895 USD  02/12/2010 7648 USD  10/11/2015 986 MKD  01/02/2013 2374 USD  22/03/2013 5601 MKD  12/03/2012 -8260 MKD  18/08/2016 4986 EUR  01/05/2012 8504 EUR  24/04/2014 8915 MKD  19/06/2011 2646 EUR  05/02/2014 -5577 MKD  30/10/2011 3444 EUR  25/01/2016 1636 MKD  18/06/2013 11238 EUR  16/07/2011 9313 EUR  24/09/2016 4282 EUR  29/08/2012 1486 MKD  04/09/2014 10940 EUR  22/11/2012 8772 USD  11/04/2014 13584 USD  10/10/2015 1508 MKD  30/08/2013 7256 EUR  19/01/2014 8962 MKD  30/09/2014 6942 MKD  19/05/2015 10280 MKD  19/03/2011 7772 EUR  20/08/2013 -8111 EUR  27/02/2010 8223 MKD  21/10/2015 12664 MKD  27/07/2011 5795 USD  08/11/2013 -5609 USD  28/09/2012 3764 MKD  24/02/2011 4864 MKD  24/08/2014 9566 EUR  30/04/2015 18170 MKD  29/05/2012 5354 MKD  20/04/2013 742 MKD  31/08/2012 7980 USD  15/10/2012 1452 MKD  06/05/2012 18068 MKD  09/02/2016 16850 MKD  28/03/2014 2684 USD  03/03/2016 142 USD  01/09/2013 2354 MKD  31/08/2014 5398 USD  30/02/2012 4733 EUR  12/07/2015 3443 MKD  02/01/2014 -6003 MKD  21/01/2012 434 EUR  15/03/2012 -7806 USD  14/10/2014 2456 USD  12/05/2010 9681 EUR  11/03/2011 -5252 USD  04/03/2010 3229 USD  30/10/2011 -2246 USD  24/12/2015 5123 USD  21/05/2010 4946 USD  09/04/2012 2182 USD  06/02/2012 -9115 MKD  13/07/2010 5058 MKD  30/10/2016 3170 USD  14/10/2016 2954 USD  11/08/2013 6056 MKD  06/12/2015 6818 EUR  16/12/2012 12244 EUR  19/07/2013 7662 MKD  11/06/2010 7092 EUR  12/10/2011 -5730 USD  03/06/2012 13366 MKD  01/05/2015 14972 EUR  18/11/2013 -4264 EUR  31/03/2014 4926 EUR  07/11/2013 7038 EUR  29/09/2016 7438 EUR  11/09/2011 -8575 EUR  20/04/2011 956 MKD  29/02/2010 -4018 EUR  22/01/2015 -1687 EUR  04/01/2015 1549 EUR  08/04/2012 5383 USD  04/02/2016 11160 MKD  20/12/2015 19272 USD  18/11/2016 -6671 USD  02/02/2016 -1362 USD  29/10/2014 -5818 MKD  29/04/2012 6022 MKD  13/04/2016 -4764 MKD  20/01/2010 13658 EUR  11/10/2010 5668 USD  24/02/2011 19528 MKD  29/08/2012 5115 EUR  04/02/2011 2778 MKD  16/12/2016 16636 EUR  21/03/2014 7324 USD  23/08/2010 3822 USD  09/02/2014 -8498 USD  05/11/2013 1516 MKD  05/10/2014 5896 EUR  28/08/2013 4728 USD  19/12/2012 -399 MKD  26/09/2011 1842 USD  09/05/2014 -2715 MKD  16/06/2011 -985 EUR  09/11/2014 -600 USD  25/07/2015 5274 EUR  27/12/2011 10602 EUR  27/08/2013 15686 USD  18/09/2014 7976 USD  30/01/2014 13864 EUR  02/05/2013 -484 USD  09/03/2011 -437 USD  23/12/2014 -2543 MKD  29/02/2012 2335 MKD  07/01/2010 -6298 EUR  24/10/2015 985 EUR  26/09/2016 6141 USD  24/11/2011 699 MKD  22/03/2016 10306 MKD  22/07/2010 1190 MKD  14/01/2011 5396 USD  01/01/2015 11420 USD  26/05/2010 8638 EUR  30/12/2013 16378 MKD  03/09/2010 -2129 EUR  06/11/2010 330 MKD  24/02/2014 17734 MKD  19/07/2014 5910 EUR  31/12/2014 18458 MKD  13/04/2013 9950 USD  13/06/2010 3188 EUR  02/01/2011 -6768 EUR  31/07/2012 -2415 MKD  28/06/2011 -849 USD  21/03/2012 15212 USD  19/08/2012 8236 USD  23/09/2010 -8351 USD  29/04/2011 17002 USD  30/05/2013 8194 USD  30/10/2014 -5994 USD  17/08/2015 2026 MKD  11/10/2014 7746 EUR  19/01/2011 3940 MKD  25/07/2013 -8370 EUR  11/04/2012 2840 EUR  29/03/2014 895 USD  17/08/2012 -9097 USD  07/06/2015 5726 EUR  25/02/2014 2418 MKD  24/05/2016 -2270 USD  17/05/2013 -7961 MKD  01/12/2012 9281 USD  17/12/2013 -1655 MKD  09/10/2011 1526 USD  20/12/2010 17276 MKD  14/02/2011 -3269 USD  25/03/2013 6310 USD  11/02/2013 2843 USD  08/07/2012 16792 MKD  03/01/2011 -2193 MKD  02/07/2011 15738 MKD  11/09/2012 8562 EUR  31/03/2014 6284 EUR  09/05/2012 5904 EUR  06/07/2016 13154 USD  16/01/2011 1242 MKD  26/07/2013 -2673 USD  31/03/2010 13804 MKD  14/02/2012 5955 MKD  20/04/2015 4901 MKD  22/03/2016 2136 EUR  31/09/2012 -6959 EUR  25/02/2015 3266 MKD | 1. 23/06/2015 6013 MKD  2. 10/12/2014 10780 MKD  3. 16/02/2015 9323 MKD  4. 28/06/2014 1419 MKD  5. 07/03/2014 7187 MKD  6. 16/10/2015 4233 MKD  7. 21/03/2014 4326 MKD  8. 12/11/2013 17082 MKD  9. 03/09/2013 6710 MKD  10. 30/06/2014 7896 MKD  11. 22/02/2015 8211 MKD  12. 15/05/2013 5652 MKD  13. 14/10/2014 8272 MKD  14. 22/03/2015 4886 MKD  15. 04/07/2013 5383 MKD  16. 06/02/2014 6221 MKD  17. 22/03/2013 5601 MKD  18. 24/04/2014 8915 MKD  19. 10/10/2015 1508 MKD  20. 19/01/2014 8962 MKD  21. 30/09/2014 6942 MKD  22. 19/05/2015 10280 MKD  23. 21/10/2015 12664 MKD  24. 30/04/2015 18170 MKD  25. 01/09/2013 2354 MKD  26. 12/07/2015 3443 MKD  27. 11/08/2013 6056 MKD  28. 19/07/2013 7662 MKD  29. 05/11/2013 1516 MKD  30. 30/12/2013 16378 MKD  31. 24/02/2014 17734 MKD  32. 31/12/2014 18458 MKD  33. 17/08/2015 2026 MKD  34. 25/02/2014 2418 MKD  35. 20/04/2015 4901 MKD  36. 25/02/2015 3266 MKD  Total: 272848 MKD | 1. 23/06/2015 6013 MKD  2. 10/12/2014 10780 MKD  3. 16/02/2015 9323 MKD  4. 28/06/2014 1419 MKD  5. 07/03/2014 7187 MKD  6. 16/10/2015 4233 MKD  7. 21/03/2014 4326 MKD  8. 12/11/2013 17082 MKD  9. 03/09/2013 6710 MKD  10. 30/06/2014 7896 MKD  11. 22/02/2015 8211 MKD  12. 15/05/2013 5652 MKD  13. 14/10/2014 8272 MKD  14. 22/03/2015 4886 MKD  15. 04/07/2013 5383 MKD  16. 06/02/2014 6221 MKD  17. 22/03/2013 5601 MKD  18. 24/04/2014 8915 MKD  19. 10/10/2015 1508 MKD  20. 19/01/2014 8962 MKD  21. 30/09/2014 6942 MKD  22. 19/05/2015 10280 MKD  23. 21/10/2015 12664 MKD  24. 30/04/2015 18170 MKD  25. 01/09/2013 2354 MKD  26. 12/07/2015 3443 MKD  27. 11/08/2013 6056 MKD  28. 19/07/2013 7662 MKD  29. 05/11/2013 1516 MKD  30. 30/12/2013 16378 MKD  31. 24/02/2014 17734 MKD  32. 31/12/2014 18458 MKD  33. 17/08/2015 2026 MKD  34. 25/02/2014 2418 MKD  35. 20/04/2015 4901 MKD  36. 25/02/2015 3266 MKD  Total: 272848 MKD |  |
|  | 234  18/05/2016 15590 EUR  20/08/2012 19264 USD  24/01/2012 9950 EUR  17/09/2014 3950 EUR  23/06/2016 19174 USD  28/01/2013 4302 MKD  27/02/2012 5638 EUR  23/02/2011 -2908 MKD  29/10/2015 8510 MKD  21/11/2011 -9359 EUR  14/02/2014 4742 EUR  11/05/2015 -3791 EUR  18/10/2014 12396 USD  19/09/2014 -7855 USD  28/12/2011 15078 EUR  18/03/2010 7770 USD  07/08/2016 -7405 MKD  27/11/2014 -4984 EUR  28/07/2014 4624 USD  08/02/2015 11576 USD  28/09/2010 8860 USD  20/04/2011 -68 USD  25/07/2015 5348 MKD  24/12/2010 -3026 MKD  06/08/2016 -7764 EUR  24/11/2016 19588 MKD  18/03/2015 7261 MKD  03/02/2014 1709 USD  25/07/2011 17390 EUR  09/04/2012 3354 EUR  20/08/2013 4641 MKD  05/11/2015 -25 MKD  03/06/2014 6461 USD  20/04/2010 17556 EUR  11/02/2016 -9598 MKD  13/08/2010 8692 MKD  23/09/2014 16714 EUR  03/05/2015 11470 EUR  15/09/2010 19356 USD  10/11/2015 3472 EUR  18/11/2016 4469 EUR  30/01/2016 842 EUR  16/10/2012 9116 MKD  26/02/2013 6498 MKD  28/01/2011 6700 USD  14/10/2015 410 USD  09/12/2013 1821 USD  05/05/2016 2376 MKD  06/05/2014 5292 USD  01/01/2012 -5101 EUR  31/02/2016 7218 EUR  01/04/2016 -5400 EUR  06/01/2016 -8033 EUR  25/11/2011 4034 MKD  24/12/2011 8090 EUR  12/04/2015 15974 MKD  08/01/2016 -5387 USD  10/12/2016 -4468 EUR  08/04/2010 3808 MKD  06/02/2016 15390 USD  04/03/2011 -2205 MKD  24/08/2014 2075 MKD  25/01/2015 12492 USD  03/08/2015 -9223 EUR  19/07/2015 -4182 EUR  30/11/2015 13738 EUR  03/05/2014 12622 USD  25/10/2011 6389 MKD  31/03/2010 2035 USD  23/01/2015 1413 USD  24/05/2013 -2531 EUR  15/05/2010 14428 EUR  02/06/2013 -3598 MKD  17/04/2011 4874 USD  29/11/2010 -5911 EUR  27/08/2011 -3049 MKD  21/09/2016 -8426 USD  11/06/2015 1108 MKD  13/09/2015 18030 MKD  03/05/2010 -3976 MKD  27/02/2010 8324 MKD  04/06/2014 10736 MKD  24/12/2010 -6261 MKD  16/04/2010 1811 USD  26/05/2015 -5433 USD  12/07/2015 -4603 MKD  02/02/2010 14942 USD  21/12/2016 -1558 EUR  29/08/2016 -5986 USD  13/09/2015 -1037 EUR  13/11/2010 5992 EUR  01/02/2014 13802 EUR  15/06/2014 1458 USD  11/08/2015 5576 MKD  17/12/2016 3745 EUR  25/05/2013 -7660 USD  02/08/2013 950 EUR  05/05/2016 -4446 EUR  21/04/2013 -7141 EUR  06/06/2016 -3336 MKD  01/02/2014 4827 MKD  16/12/2011 -3417 MKD  15/10/2015 -7082 USD  03/10/2011 -4409 USD  01/05/2010 -3315 MKD  11/10/2012 5477 USD  27/10/2012 5158 EUR  16/01/2012 19394 MKD  03/03/2013 -6981 EUR  10/12/2013 966 MKD  27/07/2016 -9371 EUR  27/04/2010 8873 EUR  04/05/2013 16118 MKD  05/08/2013 8580 MKD  07/03/2016 5960 EUR  12/09/2016 6698 EUR  22/07/2013 -3503 USD  20/04/2014 5442 EUR  25/06/2016 8782 EUR  26/04/2014 15256 USD  17/10/2014 -1597 MKD  10/03/2013 9474 EUR  31/01/2011 4978 USD  30/09/2016 840 USD  31/04/2012 740 EUR  05/05/2011 3074 MKD  08/04/2015 7126 MKD  12/07/2010 -4932 USD  17/02/2012 2152 MKD  15/03/2012 8502 EUR  03/05/2014 19514 MKD  27/03/2016 7123 MKD  25/11/2015 16996 MKD  12/04/2012 6242 USD  09/11/2015 -3477 EUR  02/02/2010 956 USD  04/01/2010 1388 MKD  04/10/2015 17870 EUR  21/02/2012 -1387 EUR  23/08/2016 2122 USD  15/06/2015 9687 MKD  28/01/2012 8917 EUR  01/09/2010 -5152 EUR  06/10/2011 7339 EUR  24/03/2012 179 USD  06/03/2016 -9364 USD  03/02/2015 6915 MKD  26/01/2016 8450 MKD  26/12/2016 -7257 USD  17/07/2016 2497 MKD  19/12/2016 6194 EUR  13/05/2010 -534 MKD  17/04/2014 3049 USD  16/01/2013 12830 EUR  22/02/2013 4648 EUR  09/02/2014 19792 EUR  15/12/2016 -362 MKD  10/06/2010 19398 USD  12/03/2010 8584 USD  23/12/2014 10280 MKD  14/03/2012 -4985 MKD  12/04/2014 -3995 MKD  06/04/2013 14658 MKD  02/07/2013 528 USD  13/10/2010 14436 USD  13/06/2015 586 MKD  08/04/2015 7732 EUR  06/06/2012 -9397 MKD  07/05/2016 -3882 EUR  15/06/2012 7340 EUR  03/05/2014 -2974 USD  29/05/2016 -1826 MKD  18/01/2016 8452 USD  21/11/2012 -2535 EUR  20/03/2015 7440 MKD  03/03/2014 4778 EUR  08/01/2016 4542 USD  09/08/2010 7726 EUR  20/03/2013 14618 MKD  01/09/2011 -9556 EUR  31/10/2010 9379 EUR  04/03/2014 6632 MKD  15/02/2013 -8381 EUR  21/06/2010 5609 USD  13/12/2011 -6389 MKD  16/12/2011 18786 MKD  30/03/2016 5292 EUR  26/03/2014 -8206 MKD  03/04/2011 15958 EUR  01/08/2013 7674 EUR  07/10/2012 -1879 MKD  21/05/2013 1243 MKD  15/07/2012 4665 USD  29/10/2012 5338 USD  11/03/2013 9441 MKD  16/06/2012 4061 EUR  21/09/2013 6960 EUR  11/01/2013 8606 USD  15/12/2013 19314 EUR  27/07/2013 5224 MKD  23/10/2016 6386 USD  09/07/2015 -234 MKD  08/03/2010 19472 USD  30/11/2016 9624 EUR  15/05/2016 19756 EUR  01/03/2010 14866 USD  12/01/2012 -5168 USD  06/02/2016 -4680 MKD  07/08/2016 18386 USD  02/02/2016 1595 MKD  25/05/2015 18856 EUR  07/06/2013 -9952 MKD  10/11/2014 9742 USD  02/08/2015 -8125 MKD  12/11/2016 1836 MKD  11/12/2012 19562 USD  05/09/2016 -6880 USD  04/02/2014 3437 MKD  28/12/2015 4514 USD  04/01/2014 4584 EUR  21/11/2013 -5318 MKD  14/02/2015 -3176 EUR  31/03/2010 6490 MKD  25/10/2014 -8143 USD  07/01/2016 11126 MKD  20/01/2015 14598 MKD  12/04/2014 -4657 USD  12/03/2015 5570 MKD  23/09/2010 -4343 EUR  29/07/2010 -8210 EUR  18/09/2013 7673 EUR  26/08/2014 3066 USD  09/01/2010 18026 EUR  26/03/2016 -5025 USD | 1. 28/01/2013 4302 MKD  2. 29/10/2015 8510 MKD  3. 25/07/2015 5348 MKD  4. 18/03/2015 7261 MKD  5. 20/08/2013 4641 MKD  6. 26/02/2013 6498 MKD  7. 12/04/2015 15974 MKD  8. 24/08/2014 2075 MKD  9. 11/06/2015 1108 MKD  10. 13/09/2015 18030 MKD  11. 04/06/2014 10736 MKD  12. 11/08/2015 5576 MKD  13. 01/02/2014 4827 MKD  14. 04/05/2013 16118 MKD  15. 05/08/2013 8580 MKD  16. 08/04/2015 7126 MKD  17. 03/05/2014 19514 MKD  18. 25/11/2015 16996 MKD  19. 15/06/2015 9687 MKD  20. 03/02/2015 6915 MKD  21. 23/12/2014 10280 MKD  22. 06/04/2013 14658 MKD  23. 20/03/2015 7440 MKD  24. 20/03/2013 14618 MKD  25. 04/03/2014 6632 MKD  26. 21/05/2013 1243 MKD  27. 11/03/2013 9441 MKD  28. 27/07/2013 5224 MKD  29. 04/02/2014 3437 MKD  30. 20/01/2015 14598 MKD  31. 12/03/2015 5570 MKD  Total: 272963 MKD | 1. 28/01/2013 4302 MKD  2. 29/10/2015 8510 MKD  3. 25/07/2015 5348 MKD  4. 18/03/2015 7261 MKD  5. 20/08/2013 4641 MKD  6. 26/02/2013 6498 MKD  7. 12/04/2015 15974 MKD  8. 24/08/2014 2075 MKD  9. 11/06/2015 1108 MKD  10. 13/09/2015 18030 MKD  11. 04/06/2014 10736 MKD  12. 11/08/2015 5576 MKD  13. 01/02/2014 4827 MKD  14. 04/05/2013 16118 MKD  15. 05/08/2013 8580 MKD  16. 08/04/2015 7126 MKD  17. 03/05/2014 19514 MKD  18. 25/11/2015 16996 MKD  19. 15/06/2015 9687 MKD  20. 03/02/2015 6915 MKD  21. 23/12/2014 10280 MKD  22. 06/04/2013 14658 MKD  23. 20/03/2015 7440 MKD  24. 20/03/2013 14618 MKD  25. 04/03/2014 6632 MKD  26. 21/05/2013 1243 MKD  27. 11/03/2013 9441 MKD  28. 27/07/2013 5224 MKD  29. 04/02/2014 3437 MKD  30. 20/01/2015 14598 MKD  31. 12/03/2015 5570 MKD  Total: 272963 MKD |  |



# 28.

Define a struct WorkingWeek that will store the number of working hours for each working day in a week (5 days) and number of week. Then define a struct Worker that will store name (char array 50) and array of working weeks (max 4).

Implement the following functions:

* int maxWeek(Worker \*w) that for the passed argument will return the number of the week with max working hours
* void table(Worker \*w, int n) that for the array of workers will print a table in the following format:

**Example**

Wor 1 2 3 4 Total

Brianna 192 238 226 225 881

Ava 236 231 184 212 863

Matthew 236 235 219 243 933

Alyssa 240 254 217 228 939

Emma 247 205 204 242 898

where in each row the name of the worker is printed, and the totals of working hours for each working week, as the sum of all totals for all weeks. A tabulator\t is used as a separator inside the row.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 10  Andrew  44 55 48 34 30  37 35 31 55 59  36 55 57 37 32  57 45 60 42 46  Mia  58 58 56 57 51  56 55 44 56 38  58 34 36 50 45  37 51 45 31 45  Natalie  50 59 42 33 43  44 30 48 38 59  53 34 53 52 31  40 45 44 41 59  John  37 43 39 59 49  51 60 49 60 33  40 38 58 39 46  42 33 31 42 60  Brianna  60 51 52 57 40  55 31 55 34 59  32 58 32 49 40  38 41 50 48 35  Samantha  40 35 45 55 38  39 31 56 55 40  55 46 38 36 32  46 31 56 47 57  Samantha  38 40 54 57 45  31 33 39 47 55  32 49 46 33 36  30 51 44 39 41  Ashley  32 39 49 31 38  45 54 41 45 55  41 59 47 35 30  45 43 40 37 53  Jacob  44 48 51 45 60  56 48 31 54 35  48 50 38 30 32  55 41 60 53 57  John  50 46 59 59 42  44 31 35 37 39  60 58 45 60 60  52 35 55 37 37 | TABLE  Wor\t1\t2\t3\t4\tTotal  Andrew\t211\t217\t217\t250\t895  Mia\t280\t249\t223\t209\t961  Natalie\t227\t219\t223\t229\t898  John\t227\t253\t221\t208\t909  Brianna\t260\t234\t211\t212\t917  Samantha\t213\t221\t207\t237\t878  Samantha\t234\t205\t196\t205\t840  Ashley\t189\t240\t212\t218\t859  Jacob\t248\t224\t198\t266\t936  John\t256\t186\t283\t216\t941  MAX WEEK OF WORKER: Samantha  4 | TABLE  Wor\t1\t2\t3\t4\tTotal  Andrew\t211\t217\t217\t250\t895  Mia\t280\t249\t223\t209\t961  Natalie\t227\t219\t223\t229\t898  John\t227\t253\t221\t208\t909  Brianna\t260\t234\t211\t212\t917  Samantha\t213\t221\t207\t237\t878  Samantha\t234\t205\t196\t205\t840  Ashley\t189\t240\t212\t218\t859  Jacob\t248\t224\t198\t266\t936  John\t256\t186\t283\t216\t941  MAX WEEK OF WORKER: Samantha  4 |  |
|  | 20  Abigail  45 49 34 37 45  51 48 52 38 50  56 60 55 36 32  54 45 44 44 32  Jacob  36 30 32 44 42  36 54 38 38 46  45 58 36 32 36  38 50 54 44 46  Matthew  48 60 59 43 31  40 34 57 39 30  44 60 40 39 42  50 32 38 40 36  William  48 59 34 53 37  44 34 48 44 40  30 50 34 43 30  49 47 38 37 31  Abigail  47 41 49 50 33  36 31 50 34 36  44 34 45 59 55  52 46 30 50 39  John  59 44 40 42 31  31 54 44 41 36  44 42 54 47 50  40 60 32 39 41  Matthew  34 36 54 36 30  47 47 58 55 33  31 40 46 59 39  43 50 60 38 55  Ashley  35 41 43 37 59  31 50 43 38 54  32 33 30 53 54  45 46 32 33 60  Natalie  47 51 41 46 57  42 59 40 42 31  52 40 58 50 31  56 35 39 50 38  Emily  35 51 37 32 41  37 32 54 54 51  34 33 38 38 42  30 37 47 30 39  Madison  46 55 37 50 32  53 39 54 60 59  54 51 52 57 53  58 48 35 55 34  Chloe  53 51 41 53 50  41 44 44 44 50  57 35 56 53 46  47 35 33 38 55  Jonathan  58 41 59 53 44  44 47 43 53 39  31 56 35 48 52  57 57 54 60 41  Isabella  57 55 34 47 54  36 38 45 52 53  40 51 47 38 48  57 36 33 30 48  Jacob  58 37 37 40 35  34 44 56 54 49  39 52 46 40 60  53 35 53 34 45  Natalie  47 50 54 42 41  32 36 53 50 52  46 50 45 50 33  34 31 39 32 32  William  44 31 50 43 53  54 44 55 45 48  47 43 50 55 47  56 49 55 59 46  Nicholas  47 59 32 48 43  58 53 59 59 60  43 45 35 53 58  39 34 59 35 34  Michael  32 45 60 50 52  55 38 41 47 30  46 30 35 42 56  38 40 38 57 45  Emily  55 36 52 36 40  45 40 47 35 42  49 34 51 32 35  42 48 43 59 51 | TABLE  Wor\t1\t2\t3\t4\tTotal  Abigail\t210\t239\t239\t219\t907  Jacob\t184\t212\t207\t232\t835  Matthew\t241\t200\t225\t196\t862  William\t231\t210\t187\t202\t830  Abigail\t220\t187\t237\t217\t861  John\t216\t206\t237\t212\t871  Matthew\t190\t240\t215\t246\t891  Ashley\t215\t216\t202\t216\t849  Natalie\t242\t214\t231\t218\t905  Emily\t196\t228\t185\t183\t792  Madison\t220\t265\t267\t230\t982  Chloe\t248\t223\t247\t208\t926  Jonathan\t255\t226\t222\t269\t972  Isabella\t247\t224\t224\t204\t899  Jacob\t207\t237\t237\t220\t901  Natalie\t234\t223\t224\t168\t849  William\t221\t246\t242\t265\t974  Nicholas\t229\t289\t234\t201\t953  Michael\t239\t211\t209\t218\t877  Emily\t219\t209\t201\t243\t872  MAX WEEK OF WORKER: Madison  3 | TABLE  Wor\t1\t2\t3\t4\tTotal  Abigail\t210\t239\t239\t219\t907  Jacob\t184\t212\t207\t232\t835  Matthew\t241\t200\t225\t196\t862  William\t231\t210\t187\t202\t830  Abigail\t220\t187\t237\t217\t861  John\t216\t206\t237\t212\t871  Matthew\t190\t240\t215\t246\t891  Ashley\t215\t216\t202\t216\t849  Natalie\t242\t214\t231\t218\t905  Emily\t196\t228\t185\t183\t792  Madison\t220\t265\t267\t230\t982  Chloe\t248\t223\t247\t208\t926  Jonathan\t255\t226\t222\t269\t972  Isabella\t247\t224\t224\t204\t899  Jacob\t207\t237\t237\t220\t901  Natalie\t234\t223\t224\t168\t849  William\t221\t246\t242\t265\t974  Nicholas\t229\t289\t234\t201\t953  Michael\t239\t211\t209\t218\t877  Emily\t219\t209\t201\t243\t872  MAX WEEK OF WORKER: Madison  3 |  |
|  | 5  Brianna  35 33 42 42 40  31 57 52 54 44  56 32 51 52 35  43 39 34 54 55  Ava  44 47 44 50 51  30 46 59 36 60  39 32 35 31 47  44 38 41 46 43  Matthew  51 37 39 55 54  36 60 54 38 47  41 57 41 39 41  56 34 43 54 56  Alyssa  59 55 57 37 32  60 54 50 37 53  54 38 34 36 55  57 55 36 41 39  Emma  58 54 42 49 44  30 45 47 44 39  52 39 44 32 37  40 50 44 56 52 | TABLE  Wor\t1\t2\t3\t4\tTotal  Brianna\t192\t238\t226\t225\t881  Ava\t236\t231\t184\t212\t863  Matthew\t236\t235\t219\t243\t933  Alyssa\t240\t254\t217\t228\t939  Emma\t247\t205\t204\t242\t898  MAX WEEK OF WORKER: Matthew  4 | TABLE  Wor\t1\t2\t3\t4\tTotal  Brianna\t192\t238\t226\t225\t881  Ava\t236\t231\t184\t212\t863  Matthew\t236\t235\t219\t243\t933  Alyssa\t240\t254\t217\t228\t939  Emma\t247\t205\t204\t242\t898  MAX WEEK OF WORKER: Matthew  4 |  |