NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD

Object Oriented Programming (CS1004)

SPRING 2022 ASSIGNMENT # 1

Due Date ≈ Friday, March 11th, 2022 (11:59 pm)

Instructions

Submission: Combine all your work in one .zip file. Use proper naming convention for your submission file. Name the .zip file as **SECTION_ROLL-NUM_01.zip** (**e.g. P_21i0412_01.zip**). Your zip file should not contain any folders or subfolders. It should only contain .cpp files for each question, e.g. Q1.cpp, Q2.cpp, Q3.cpp, Q4.cpp, Q5.cpp OR if additional files are asked they will be mentioned with each question. Submit .zip file on Google Classroom within the deadline. Failure to submit according to the above format would result in **25% marks deduction**. Submissions on the email will not be accepted.

Plagiarism: Plagiarism cases will be dealt with strictly. If found plagiarized, both the involved parties will be awarded zero marks in this assignment, all the remaining assignments, or even an **F grade** in the course. Copying from the internet is the easiest way to get caught!

Deadline: The deadline to submit the assignment is 11th **March 2022 at 11:59 PM**. Late submission with marks deduction will be accepted according to the course policy shared earlier. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

Bonus: In case you implement any additional feature which you think is worth of bonus, make it prominent so that we can see it at runtime.

Note:

- *Each question will be graded on the basis of your effort, additional marks will be awarded for using good programming practices, including: memory efficient programs, well-written, good design and properly commented.*
- All programs must be generic.
- You can change the argument, return type and also add new data members in the given structures.
- Follow the given instructions to the letter, failing to do so will result in a zero.

Problem 1 – ICE: An ice-cream shop namely ICE is selling 4 flavors to customers in 5 cities. There are 10 outlets in each city. The customers are provided with an ID and top 10 regular customers get discount of 5% on family pack from each outlet. Suppose a customer can only visit one outlet per day. Regular customers become eligible for discount if they purchase ice-creams of minimum Rs. 2000 within a week and discount offer remains valid for the next week. Week begins from Sunday and ends on Saturday. Customers are sorted in descending order of their total purchasing worth. ICE calculates the total profit at the end of each month. Total production cost including tax is 70% of the selling price. Write a program to cover all details of ICE.

A table with prices is provided as:

Flavor	**Size**	**Price**
Mango	Mini	Rs.30/-
	Family Pack	Rs.120/-
Strawberry	Mini	Rs.40/-
	Family Pack	Rs.130/-
Chocolate	Mini	Rs.40/-
	Family Pack	Rs.130/-
Vanilla	Mini	Rs.25/-
	Family Pack	Rs.110/-

Add any number of data members in the basic layout provided below or make more structures.

1. Make a function to calculate weekly purchase by a customer.
2. Make a function to print the names of top 10 customers in each outlet.
3. Provide a function to calculate and print the revenue generated by each outlet.
4. Provide a function to identify and print the age group more likely to receive the discounts.

All the outputs should be displayed in a well-structured manner (e.g. tables).

[Hint]: Consider each iteration as a single day. You can trigger purchases in each outlet randomly.

```
struct Address{
   char* City;
   char* Location;
   …
};
struct Customer{
   string name;
   int custID;
   int purchase;
   int age;
   …
};
```

```
struct Flavor{
   int flavorID;
   int Price[2];
   …
};
struct ICEOutlet
{
    int OutletID;
    Address Outlet_Add;
    Customer TopCus[10];
    Flavor item[5];
   …
}
```

Problem 2 - PSL Player Draft: You have to develop a program that PSL teams will use to pick new players for the coming season. League has a total of 4 teams where each team can have a maximum of 16 players. Let's assume that N Players have already been listed in the draft, and for each we have the following information as shown in the table below.

Sr. No.	Name	Type	Batting average	Batting strikerate	Bowling average	Bowling strikerate	Value	Availability
1	Babar Azam	Batsman	50.93	130.00	-1	-1	10,00,000	true
2	Ben Cutting	Allrounder	23.08	149.73	31.45	21.0	9,00,000	true
3	Dale Steyn	Bowler	-1	-1	21.96	19.3	11,000,000	true
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
N	Imad Wasim	Allrounder	19.10	125.97	25.46	23.4	8,00,000	true

```
struct PlayerInfo{
   char* name;
   int type; //type of the player “batsman” or “
   …
};
```

```
struct Team{
   char*  teamName;
   int teamRank;
   Player* teamPlayers;
   …
};
```

You have to write atleast the following functions. Only name of the functions is given you can have any number of arguments in them:

1. playerData(): Get the number of players to enlist in the draft and store their information (as shown in the table above). For a batsman bowling average and strike rate should be set to -1, similarly for a bowler batting average and strike rate should be set to -1.
2. sortPlayers(): Reorganize the player information on the basis of their expertise. Only allocate memory required to store players' data separately (batsman bowler and allrounder).
3. assignRanks(): Assign ranking to the 4 teams on the basis of random number.
4. teamPicks(): To calculate how many new players each team can buy in the draft, the team ranked first gets to decide it first, then the 2nd and so on.
   a. Each team will be asked to tell the number of players they want to retain out of 16 (lets say this number is R). R cannot be less than 7 and greater than 11.
   b. New players they can have will be total players minus the players they want to retain (i.e. 16 – R)
5. playerSelection(): Player selection function where each team chooses their pick one by one (starting with the team ranked first), you have to ask the team which type of player they want to pick (batsman, bowler and allrounder). Once a player is picked by a team change his status to false (meaning sold).

Problem 2 - Fast Cafeteria: You have to develop a software for Fast Cafeteria. The overall objective is that the program should be able to calculate total sales, salaries of salesman, types of item sold, and record each customer data (age, gender, time and day of purchase). You have to enter data of three salesperson and at least 10 FoodItem(s). Consider following structures, you can always add additional data member if you feel it will result in better/efficient design:

```
struct Sales_Person{
   int emp_ID;
   char* emp_Name;
   double Sales;
   …
};
struct Salary{
   int fixedSalary ; //Rs. 15000
   double comission;  //2% of sale price of all sales
   …
};
struct salesData{
```

```
struct Customer{
   int cust_No;
   int C_Gender;
   int age;
   …
};

struct FoodItem{
   int id; //1-fastfood, 2-regularfood, 3-packedfood
   char* itemName;
   int price;
   …
```

```
    Customer c;                                          };
    int itemOrdered;
    FoodItem* f;
    SalesPerson s;
    …
};
```

You have to write following functions:

1. totalSales(): Total sales
2. totalSalesBySalesPerson(): Total sales done by a specific sales person
3. topSalesPerson(): List of sorted sales person along with the total sales done by them
4. commissionSalesPerson(): Calculate commission of a sales person
5. totalSalary(): Calculate the amount that the owner has to pay to all sales person
6. salesByType(): returns sales of a particular fooditem

Problem 3 - Predicting Sales in FAST Cafeteria: You are required to enhance the FAST Cafeteria application (Problem 2) to help Cafeteria improve their future sales. This will be done by a prediction algorithm which will predict an output given an input (for example, input age and output can be a particular food item). You will need to generate data of at least 30 sales. Since, each sales data consists of sales person, customer and food item(s).

One of the simplest supervised learning algorithms is Linear regression. It fits a straight line ($h(x) = \theta_o + \theta_1 x_1$) through data points. For example, in the graph below X is input variable (there can be multiple input variables) and Y is the output variable. The goal of linear regression is to fit a line through these data points which is the most representative of the data. This is done by minimizing the cost function which measures the root mean-squared error between the predicted output value – h(x) and true output value – Y. For the given h(x), this will be done by finding the best value of θ_1.

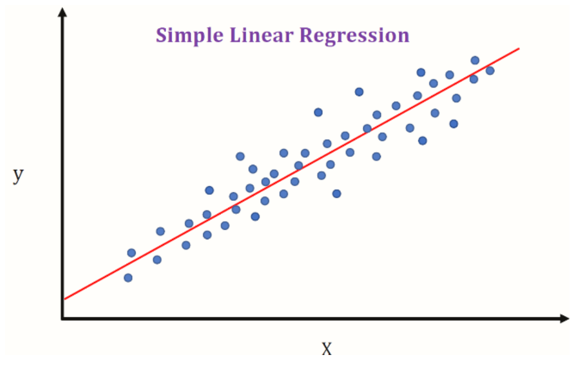

Figure: Y as a Linear function of X

Gradient descent method is an iterative algorithm for finding the local minimum of a function. In h(x), θ_j are the parameters mapping X to Y. Your goal is it to implement Gradient descent as a recursive function to find the best approximation by updating θ_j. You will keep on repeating the following until convergence. It will be done for all θ_j. For the above equation you need to do it only for θ_1 as h(x) is only a linear function of x_1.

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}, \text{(for every } j)$$

You can find more detail on gradient descent at http://cs229.stanford.edu/notes2021fall/cs229-notes1.pdf (Page 3 to 7). An example is also given in these notes, where Y – Price (is the output) and X – Living area and bedrooms (are the inputs). This shows that Y can depend on multiple inputs. So the θ_j is a two dimensional vector for the inputs Area (x1) and No. of bedrooms (x2).

In this scenario Y (output) is sales. You will display different attributes that can be picked as input (X), for example X can be: (Age) or (Age and Gender) or (fooditem) or (fooditem and Gender). You will have to display the graph showing 30 data points.

Extra Credit: For displaying the regression line through data points.

Problem 5 – Weighted Resource Constrained Scheduling Problem: One of the key tasks in Project Management is scheduling where its completion time is determined. For scheduling you at least needs a list of tasks, their durations and dependencies. One method which takes this task information to determine schedule is Critical Path Method (https://asana.com/resources/critical-path-method)

We will develop a more sophisticated method which not only considers task information but also resources and their skills when generating the schedule. Consider following structures, you can always add additional data member if you feel it will result in better/efficient design:

```
struct task{
   int id;
   int dur;
   int s_Time; //start time of each task
   int* dep; //array of tasks ID's
   int skill_ID;
   …
};

struct Skill{
   int skill_ID;
   float proficiency;
   …
};
```

```
struct project{
   int id;
   int t; //duration of project
   task* tasks;
   …
};

struct Resource{
   int res_Id;
   bool res_Availability;
   Skill res_Skill;
   …
};
```

Start time of at least one task will be zero (0). A project will have N tasks and R resources, N and R will be entered by the user. You have to implement the following functions, think about the parameters required and return type of the following functions

1. addTasks();
2. setTaskDuration();//change task duration of all tasks
3. set_nth_TaskDuration();//change duration of a specific task
4. printTaskDependencyList();//print dependencies of a specific task
5. calculateBasicSchedule(); Use Critical Path Method to calculate this schedule. print completion time of the project
6. printCriticalTasks(); returns array of critical tasks and displays them – sum of their duration should be equal to project completion time
7. completionTimeWithResources(); You can use basic schedule and depending on the resource availability a task can start.
8. completionTimeWithResourceProficiency(); You can use basic schedule and depending on the resource availability a task can start.