NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD

Object Oriented Programming (CS217)

SPRING 2021 ASSIGNMENT # 3

Due Date: Sunday, May 9th, 2021 (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 **ROLL-NUM_Section.zip** (**e.g. 20i0412_J.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, …, Q8.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 9th **May 2021 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.

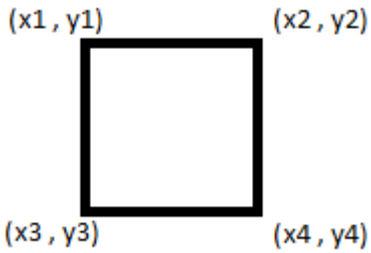
Bonus: In case you implement any additional feature which you think is worth of bonus, make it prominent so that we can see it at runtime.

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 and classes.
- Follow the given instructions to the letter, failing to do so will result in a zero.

Question 1: Overlapping rectangles – A rectangle is identified by its four coordinates, as shown in the figure below. Make a structure to hold data for every rectangle, there is bare minimum information that you to store for each rectangle i.e. its coordinates. You can also improvise and make additional members in the structure if you think it will help in doing any of the following tasks. Write the following functions:

1. Generate n number of rectangles
2. Find and return the area of all the rectangles, given the rectangles in the above format
3. Sort and return all rectangles in the ascending order based on their area.
4. Find the set of rectangles that overlap the largest rectangle
5. Write a function main (as a controller) to call the above functions in the correct order to identify the set of rectangles that overlap the largest rectangle



Question 2: Smart mirror – You customize smart mirrors for your customers. For each customer the options that they may want to display on smart mirror may include: upcoming Tasks, Time, Weekly quotes and Date. Your task is to write a C++ program using the knowledge of structures and OOP that will be able to customize display for any user.

1. Display DATE in day/month/year or "day Month, Year" e.g. 23rd April,2021, format according to his choice
2. Display a weekly quote section, one that he can edit anytime.
3. Display and be able to set the time according to 24 hour clock and 12 hour clock format.
4. Display a section of upcoming tasks, and keep an option for adding tasks or deleting a task from the mirror.
5. 

Remember all these tasks should be accessible only through the mirror. It is not required that you display the tasks in the manner shown below; however the options should cater for all possible user choices. Also note, that in order to avoid clutter on the mirror, you can display a new screen for choosing to edit, add, and exit back to the main mirror screen.

Question 3: Bookshop – You have to develop a software for a bookshop. The overall objective is that the program should be able to calculate total sales, salaries of salesman, types of book sold, and record each customer data. You have to enter data of three salesperson and at least 10 books (try to enter data of three different types of books History, Science fiction, and Adventure). Consider following structures, you can always add additional data member if you feel it will result in better/efficient design:

```
struct Sales_Person{//to store data of each sales person
   int emp_ID;
   char* emp_Name; //duration of project
   double Sales;
   Customer* emp_Customer_Array; //array of customers that salesperson sold books to
   Salary emp_sal; //salary of the employee
};

struct Salary{
   double fixedSalary ; //every salesperson has a fixed salary for a daily wager Rs. 15000/-
   double comission; //2% of sale price of all the books sold

};

struct Customer{
   int cust_No;
   char* cust_Name; //duration of project
   Address cust_Address; //structure itself
   Book bk; //
};

struct Book{
   int id;
   string type
   char* bookName;
   char* authorName;
   double price;
};
```

You have to write following functions:

1. totalSales(): Total sales done
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. booksInventory(): remaining books in inventory
7. booksInventoryByType(): remaining books by type

Question 4: Simulate Court Piece Card Game – In this game there are four players in fixed partnerships, partners sitting opposite. Cards are dealt and game is played anticlockwise. Make a structure A standard pack has four suits, the cards in each suit are ranked from high to low A-K-Q-J-10-9-8-7-6-5-4-3-2.

Description of function is taken from this link: https://www.pagat.com/whist/rang.html. For more understanding of rules it is recommended.

You have to write following functions to simulate the game:

1. chooseDealer(): Out of the four players one is chosen randomly to deal the cards for the first time (referred as "dealer"). The person to dealer's right is the one who announces trump suit and is called "trump-caller".
2. Deal_and_Tumps():The dealer deals a batch of five cards to each player (five cards for each player would be chosen randomly from the available cards). The trump-caller player looks at his or her five cards and (without communication with any other player) chooses and announces the trump suit. Then the dealer deals out all the remaining cards in batches of four (13 cards per player).
3. revealCards(): Although, this is not done in the game as only the player knows his/her cards but this function is just to ensure that cards have been dealt correctly
4. oneTrick(): The objective is to win the trick, so you have to carefully implement an algorithm that chooses the card with the potential to win the trick (ranking of the cards have already been mentioned). The player to dealer's right leads any card to the first trick. All the other players must follow suit. When all four players have contributed a card the player of the highest card of the suit that was led wins the trick unless one or more cards of the trump suit were played, in which case the highest trump wins. The player who won the trick leads any card to the next trick.
5. game(): The object of the game is to score **courts** by winning the majority of the tricks (hands). This will be called trick function repeatedly unless a team has won 7 tricks. In case the other team has not been able to win even a single trick the game will continue so that the former team may end up winning all 13 tricks.
6. switchDealer(): The dealer is always a player from the team that lost the previous deal, so that the winners of the previous deal call trumps.

For a detailed description of game rules please see (https://www.pagat.com/whist/rang.html)

Consider following structures, you can always add additional data member if you feel it will result in better/efficient design:

```
struct player{
   int id;
   int t; //duration of project
   cards* cardsInHand;
};


struct card{
   char i;
   string suit;
};
```

Question 5: Table Tennis Fantasy League - You have to develop a table tennis fantasy league. Here are the details of league: no. of teams (4) and no. of player per team (3). Each team will have to play the other team twice, so you can calculate the total number of matches. Consider following classes, you can always add additional data member if you feel it will result in better/efficient design:

```
class Player{
  private:
      char* name;
      string type; //type of the player "attacking" or "defensive"
      double points; //randomly assign points between 30 to 40 randomly

  public:
    // constructors
};

class Team{
  char*  teamName;
  int teamRank;
  Player* teamPlayers;
};

class Match{
  int Match_No;
  Team homeTeam;
  Team awayTeam;
};
```

You have to write at least the following functions. Also decide which functions will become member functions of which class. Only names of the functions are given you can have any number of arguments in them:

1. addPlayers(): Add players data so that they are available to be picked by teams.

Sr. No.	Name	Type	Points	Availability
1	Name Player 1	Attacking	35.93	true
2	Name Player 2	Defensive	33.08	true
3	Name Player 3	Attacking		true
.	.	.	.	.
.	.	.	.	.
15	Name Player 15	Attacking	39.10	true

2. returnPlayers(): This function is supposed to return player of certain type (attacking or defensive).
3. assignRanks(): Assign ranking to the 4 teams on the basis of random number.

4. teamSelection(): Form the team on the basis of strategy, a defensive team will prefer two defensive players and one attacking player. Whereas an attacking team will prefer two attacking players and one defensive player.
5. playerSelection(): Player selection function where each team chooses their pick one by one. Once a player is picked by a team change his status to **false** (meaning sold).
6. generateMatchStats(): This function will generate match status. Assign a winning team players' point between 10 and 20 randomly. For the losing team assign them points between 5 and 10 randomly.
7. bestPlayersInMatch(): Display the information of the best player of both the teams
8. UpdatePlayerPoints(): After a match update ranking points of players
9. printLeaderBoard(): Prints the sorted list of player in all teams
10. printTeamRanks(): Prints the ranking of teams

Question 6: Plane Landing problem – Plane landing is an optimization problem and need sophisticated algorithms to assign landing time and runway to each plane. You have to develop a very simplistic solution to the plane landing problem. Here is the assumption for the problem that you are going to solve:

Input to the program is a schedule of flights who are going to land in 10 minutes. For each flight you will have to enter some following information: flight number, remaining fly time (this time is updated when it contacts the AirController), scheduled landing time (this time of all the flights have to be within this 10 minute window). Sample data might look like this:

Arrival Time	From	Flight No.
17:20 17-Apr-2021	Riyadh	PA 2755
17:25 17-Apr-2021	Quetta	PK 352
17:30 17-Apr-2021	Karachi	PA 204

Airport for which we are designing this solution has only three runways. Each runway can be used for landing. Once a plane is allocated to a runway it will be occupied for at least 2 minutes.

Consider following classes/structures, you can always add additional data member if you feel it will result in better/efficient design:

```
class AirController {
  private:
    // think about the private data members...
  public:
    // constructors

    // provide definitions of following functions...
    /*Special class with only one instance during execution of the program, there is only air traffic
```

```
    controller and landing time is allocated to plans in a centralized fashion*/
    RunwayStatus();//prints the status of all the runways
    FlightContact(); //prints the data of flights who have contacted the controller
    AssignRunway();//assigns runway to a flight
};


struct Runway {
    int id;
    string type;
    status availability;
    int dur;
    int s_Time; //start time of each task
};

class Flight{
  private:
    // think about the private data members...
    string flight_No; //
    double arrivalTime; //arrival time which is known at the departure.
    double remaingFlyTime; //how long can the fuel last it no runway is available
    double  scheduled_landing_Time; //
    Runway rn; //

  public:
    // constructors

    // provide definitions of following functions...
    addFlight(); //function to add flight information
    UpdateRemainingFlyTime(); //This is updated once the flight contacts AirController
    UpdateLandingInfo(); //get landing time from the AirController depending on the availability of the
    runway

};
```

Question 7: Scheduler – An organization can potentially work on multiple projects at any given time. Each project consists of multiple tasks, which is the work that is to be performed in the project. You have to implement a class which takes the task information from the user for a project. Then calculate completion time of the project.

PART A: Calculate the completion time of the project given the task information. How to determine project completion time? This link may be helpful: https://www.workamajig.com/blog/critical-path-method

Consider following structures, you can always add additional data member if you feel it will result in better/efficient design:

```
class task{
  private:
      int id;
      int dur;
      int s_Time; //start time of each task
      int e_Time; //end time of each task
      int* dep; /*list of predecessors of this task - To simplify we assume that a higher number task will
      depend on a lower number task e.g. T2 can depend on T1 OR T4 can depend on T2 but the
      opposite is not true.*/
      char skill; //required for PART B
      Resource res; //required for PART B

  public:
  addTasks();
  setTaskDuration();//change task duration of all tasks
  set_nth_TaskDuration();//change duration of a specific task
  printTaskDependencyList();//print dependencies of a specific task

};

class project{
  private:
      int id;
      int t; //duration of project
      task* tasks;
  public:
    // provide definitions of following functions...
    Project();// default constructor
    Project(task* ts, int n);//initialized the project with n tasks

    //you have to implement the following functions
    // think about the parameters required and return type of the following functions
    completionTime();//print completion time of the project
    printCriticalTasks();/*returns array of critical tasks and displays them – sum of their duration
    should be equal to project completion time*/
    completionTimeWithResources();//for PART B
};
```

PART B: Consider that now you have to make the same schedule but also have to pick resources and assign to the tasks. Each task has a special skill required for its completion (we will use 4 skill types ‘A’, ‘B’, ‘C’, and, ‘D’). Now you will have to generate completion time by assigning resources to the tasks. If a resource is not available, then the project will be delayed. Enter data for 10 resources using the following structure.

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