Lab6: Structures

Instructions

- *Make your own file named Submission.cpp . Please don't include the main function while submitting the file.*
- *Data member names of each structure should be the same as per mentioned in each question.*
- Please read the questions carefully, read them twice even thrice to understand them completely.
- Please see the corresponding function signature in the skeleton file to further understand the question.
- In case of any query, please raise your hands and we will be there to solve your query.
- Please concentrate, understand and code. Good Luck :)

TASK 1

(a) A Student identity card number can be thought of three parts: the campus code(I), the batch number (2016), and the ID (0392). Write a structure named as **StudentCard** to store these three parts of Student identity number. Data members should be named as **campusCode** of type char, **batch** of type int and **ID** of type int .

(b) Write a function StudentCardArrayIntialize which takes an array *arr* of type StudentCard, of size *s*. You have to store StudentCard objects at each index of array arr. Assign values to each object of StudentCard such that students are placed in ascending order of its student IDs and batch. It continues for all objects sequentially of size s. For example : the student of batch 2015 is placed in ascending order before the students of batch 2016.
Hint:You can use random numbers to generate the 4 digit ID.

void StudentCardArrayIntialize(StudentCard arr[],int s)

(c) Write a function PrintStudentCard which takes s as StudentCard object ,you have to print s in following format
(campus code) Batch-ID .
Example: (I) 2015-1011

void PrintStudentCard(StudentCard s)

TASK 2

(a) Declare a structure named as **CustomTime** having three data members named as ; **hours** of type int, **min** of type int and **seconds** of type int.

(b) Write a function timeToSeconds which takes CustomTime object t1 as function parameter, Calculate the total seconds in time and convert this object t1 to seconds (of type long) and return it.
long timeToSeconds(CustomTime t1)

(c) Write a function SecondsToTime which takes long t as function parameter and return it in the form of a CustomTime object.
CustomTime SecondsToTime(long t)

(d) Write a function AddTimes which takes as input parameters CustomTime object t1 and CustomTime object t2

1. You have to convert these two objects t1, t2 in seconds using above defined function timeToSeconds.
2. Add both seconds returned from above step.
3. Use above defined function SecondsToTime and convert back it to the CustomTime object and return it.

CustomTime AddTimes(CustomTime t1, CustomTime t2)

(e) Write a function MakeNewTime which only create a dynamic memory of size 1 on heap for CustomTime object and return its pointer to CustomTime.
CustomTime* MakeNewTime()

(f) Write a function IntilaizeTime which takes CustomTime* p and int totalSec as function parameters, convert totalSec into hours, minutes and seconds and assign it to a p using dynamic memory allocation.
Hint: use above defined function MakeNewTime and SecondsToTime
void IntilaizeTime(CustomTime* & p,long totalSec)

(g) Write a function MakeArrayOfTimes which only create a dynamic array of size s on heap for CustomTime object and return its pointer to CustomTime.
CustomTime* MakeArrayOfTimes(int s)

(h) Write a function IntilaizeTimeArray which create a dynamic array of size s on heap using previously defined function MakeArrayOfTimes, assign it to p of type pointer to CustomTime and initialize its values from the array of hours, mins and sec. Each array is of size s and type int.
void IntilaizeTimeArray(CustomTime* &p,int hours[],int mins[],int sec[],int s)

TASK 3

(a) Write a structure **Fraction** that has the following data members: int deNom, int Nom. where **deNom** is the denominator value and **Nom** is the nominator value

(b) Write a function which takes Fraction f as reference of fraction object, int deNom and int Nom as function parameters. Your task is to initialize object f with values of deNom and Nom.
void IntializeFraction(Fraction &f,int deNom,int Nom)
Hint: you have to check for zero as denominator

(c) Write a function which takes Fraction f1 and Fraction f2 as function parameters . Your task is to sum the two fractions f1 and f2 and return its sum as a new fraction.
Fraction AddFractions(Fraction f1, Fraction f2)
Addition: a/b + c/d = (a*d + b*c) / (b*d)

(d) Write a function which takes Fraction f1 and Fraction f2 as function parameters . Your task is to multiply the two fractions f1 and f2 and return its result as a new fraction.
Fraction MultiplyFractions(Fraction f1, Fraction f2)
Multiplication: a/b * c/d = (a*c) / (b*d)

(e) Write a function which takes Fraction f1 and Fraction f2 as function parameters. Your task is to subtract the fraction f2 from fraction f1 and return its result as a new fraction.(f1-f2)
Fraction SubtractFractions(Fraction f1, Fraction f2)
Subtraction: a/b - c/d = (a*d - b*c) / (b*d)

(f) Write a function which takes Fraction f1 and Fraction f2 as function parameters. Your task is to divide the fraction f1 by fraction f2 and return its result as a new fraction.
Hint: you have to check for zero as denominator
Fraction DivideFractions(Fraction f1, Fraction f2)
Division: a/b / c/d = (a*d) / (b*c)

TASK 4

(a) Write a structure `Student` that has the following data members with private access modifier: rollNo of type int, name of type char*, city of type string and phone of type string.

(b) Write a member function that initializes all data members of `Student` Structure. You have to assign default values using `Initialize()` such that rollNo assigned with 0, name with dynamic allocation on heap of size 25, city to Null string and phone to Null string.
`void Initialize()`

(c) Write a member function `Destroy()` that deallocates all the memory allocated by `Initialize()` member function.
`void Destroy()`

(d) Write get and set member functions for each data member of `Student` structure. Name of getter and setter functions must be like getRollNo, getCity, setCity.
You have to validate inputs, such that roll no should always be greater than 1000, name length should be greater than 4 letters, while city and phone can not be Null.

```
bool setRollNo(int r)
int getRollNo()
bool setName(char* n)
char* getName()
bool setCity(string c)
string getCity()
bool setPhone(string c)
string getPhone()
```

(e) Write a global function `SetStudentArray` which takes five arguments, an array of type `Student` named `arr`, int `size`, array of char pointers as `names`, array of city as `cities` and array of phone numbers as `phones` . You are required to assign values to each `Student` in `arr` using above defined setter functions. Note* `setRollNo(int r)` validates that rollNo of first student should start from 1001 , second student rollNo would be 1002 up to so . Initialize name, city and phone from the arrays passed as arguments
`void SetStudentArray(Student arr[], int size, char* names[], string cities[], string phones[])`

(f) Write a global function `GetStudentArray` which takes two arguments, an array of type `Student` named `arr` and int `size`. You are required to get and display information of each `Student` in `arr` using above defined getter functions.

```
void GetStudentArray(Student arr[], int size)
```

TASK 5

(a) Write a program to define a structure `Employee` that has the following data members empNo as int, basicPay as long, houseRent as long, medicalAllow as long, conveyanceAllow as long, netPay as long.

(b) Compute the allowances based on basic pay and Allowances are computed as:

- House rent is 54% of the basic pay.
- Medical Allowance is 15% of the basic pay.
- Conveyance allowance is 20% of the basic pay
- Compute net pay

You have to define following member functions in `Employee` structure.

```
void setBasicPay(long bp)
long getBasicPay()
void setEmpNo(int e)
int getEmpNo()
void calculateHouseRent()
void calculateMedicalAllowance()
void calculateConveyanceAllowance()
void calculateNetPay()
long gethouseRent()
long getMedicalAllowance()
long getConveyanceAllowance()
long getNetPay()
```

(c) Write a function named as `Swap` that takes two parameters of type Employee: emp1 and emp2. You are required to swap values of these parameters.
*Note you are not allowed to copy complete object using assignment operator(=) rather you have to copy each data member separately into its corresponding data member.

```
void Swap(Employee & emp1, Employee & emp2)
```

TASK 6

(a) Define a structure to represent a bank account named as `BankAccount` that includes the following members:
`Private Data Members:`

- `depositorName` of type string.
- `accountNumber` of type string
- `accountCat` of type `accountCategory` (having data members, int Id, string Name)
- `Balance` of type long

`Public member functions:`

- `void Initilaize(string dp, string an, int acountTypeId, string accountTypeName, long balance)`

- To deposit an amount,
  `bool depositAmount(long amountToDeposite)`
  Hint: you have to validate that amount to deposit is non-negative.
- To withdraw an amount after checking the balance.Return False if amount is less than withdrawl amount otherwise return True in case of successful transaction.
  `bool withdrawAmount(long amountToWithdraw)`
- To get balance in given accountNum,
  `long getAmount()`

TASK 7

(a) Your goal in this problem is to define and use the structure(s) to store semester registration information for students at FAST. A semester registration consists of a semester code, and course registrations of between one and five courses. A course registration consists of course code, course title, credit hours, section, and repeat count.

Semester Registration	
Structure Members / Properties	**Properties Description**
Semester Code	Stores the code of semester, like "Spring 2018"
Course Registrations	Stores courseCount courses registered in the semester

Course Registration	
Structure Members / Properties	**Properties Description**
Course code	Stores course code, like "CS 103"
Course Title	Stores course title, "Computer Programming"
Credit Hours	Stores credit hours of the course
Section	Stores section in which student is registered
Repeat Count	Stores repeat count of the course for the student

After defining the structure(s), write the following global functions:

1. `GetCreditHoursCount`: receives a semester registration as parameter and returns the total number of credit hours registered in it.

2. `FindCourseInSemesterRegistration`: receives a semester registration, and a course code as parameters and returns true if the course is registered in the semester.

TASK 8

(a) Your goal in this question is to write a program to manage a ShoppingList. For this you will need to create a structure ShoppingList that should be able to store all the details of shopping items with the following functionalities:

Your ShoppingList have a fixed capacity (10) of ShoppingItems. For each ShoppingItem you will store its name, its quantity and its price. A new ShoppingItem cannot be added to ShoppingList if the list is already full

Whenever you buy a new item you will add it to the shoppinglist if the capacity is not full.

Define and use structures for the ShoppingList and ShoppingItems. Identify all the data members and member functions of these structures and write them. For instance, your structures must provide, apart from other functions, following interface (`public`) functions.

ShoppingList	
Member Function	**Description**
AddItem	Add a new shopping item to the shopping list.
Print	Should print the current list of added items to list
TotalCost	Should print the total cost of shopping list
ShoppingItem	
Member Function	**Description**
InputItem	Add details for the current item.
Display	Should display the information of item

TASK 9

(a) Your goal in this question is to write a program to manage a Parking Garage. For this you will need to create a structure Parking Garage that should be able to simulate a basic parking garage with the following functionalities:

Your garage should have a fixed capacity (5 cars at most). For each car stationed at your garage you will record its entry time (current day hour in 24h format), its registration number and its allocated slot (each car can be allocated parking slot 1 to 5). A new car cannot be parked if the garage capacity is already full.

Whenever a new car arrives at your garage you will add it to the garage if the capacity is not full. When a car leaves the garage you will ask the user current time (hour in 24 hour format) and charge him Rs. 20 per hour, and declare the slot as available so that a new car can be parked here.

Define and use a structure (or class) for the Garage. Identify all the data members and member functions of this structure (or class) and write them. For instance, your class or structure must provide, apart from other functions, following interface (`public`) functions.

Member Function	**Description**
Initialize	Sets all the member to reasonable values.
Print	Should print the currently parked cars information and empty slots information
IsFull	Should return whether Garage is full or not.
ParkCar	Should add a new car to the Garage if it is not full.
RemoveCar	Should remove a car from a given parking slot.