Genel Kurallar - General Rules

- fonksiyonların prototipleri doğru yazılmalıdır - function prototypes must be written correctly.
- fonksiyonlardan beklenen işlevler doğru çalışmalıdır - the functions should perform correctly.
- ekran çıktıları ve diğer tasarımlar tutarlı olmalıdır - screen outputs and other class designs must be consistent.
- sayı-string dönüşümünde **`to_string()`** fonk. kullanılabilir - you can call **`to_string()`** func. string to number conversion.
- Time ve Date sınıflarının `ampm` dışındaki değerleri tamsayı türünde olmalıdır - the values of Time and Date classes must be integer except `ampm` .

1) Sizden bir **Time** sınıfı tanımlamanız istenmektedir. Bu sınıfın `saat, dakika,saniye ve ampm` (öğleden önce mi sonra mı) şeklinde özellikleri bulunmaktadır. Bu değerlerin varsayılanı sırasıyla `(0,0,0,'a')` olmalıdır. `ampm` parametresi karakter tipindedir ve 'a' karakteri olduğunda öğleden önce, 'p' karakteri olduğunda öğleden sonrayı temsil etmektedir - You are asked to define a **Time** class. This class has `hour, minutes, seconds and ampm` properties. The default values of the properties must be `(0,0,0,'a')`. the `ampm` property is a character type and it means “before noon” if `ampm` is ‘a’ and “after noon” if `ampm` is ‘p’.

1.a) Bu sınıfın parametreli, parametresiz yapıcı fonksiyonları, get set fonksiyonları ile b,c,d maddelerindeki fonksiyonları içerecek şekilde sınıfı tanımlayınız. Parametreli yapıcı fonk., `saat, dakika, saniye ve ampm` bilgisini parametre olarak almalıdır. - Declare the default and parameterized constructors, getter and setter functions and the functions used in b,c,d. The parameterized constructor must take `hour, minutes, seconds and ampm` variables as parameter (5p).

```
class Time{
  public:
    Time(int _hour=0, int _min=0,int
_sec=0,char _ampm='a');
    string toString12();
    string toString24();
    Time operator + (int _min);
    int getHour();
    int getMin();
    int getSec();
    void setTimer(int h, int m, int s);

  private:
    int hour;
    int min;
    int sec;
    char ampm;
};
```

1.b) Parametreli yapıcı fonksiyonu gerçekleyiniz. Saat parametresi 12 den büyük geldiğinde otomatik olarak öğleden sonra bir saat olduğu anlaşılmalıdır, verilen parametreler negatif ya da saat olmayacak şekilde değildir - Write the parameterized constructor function code. if hour is greater than 12, it means that the time is “after noon”. Given parameters are not negative and should be accepted as valid values for any time’ (5p).

```
Time::Time(int _hour, int _min, int _sec,
char _ampm){
      this->hour  =  _hour;
      this->min   = _min;
      this->sec   = _sec;
      this->ampm  = _ampm;
      if(_hour>12){ // saat 12 den büyük ise
pm
        this->hour = _hour%12;
        this->ampm = 'p';
      }  }
```

1.c) Saati 12 saat formatında string olarak geri döndüren `toString12()` fonksiyonunu yazınız. Sabah saat 9'u çeyrek geçe için "09:15:00 AM" geri döndürmesini bekliyoruz - Write the function `toString12()`, which returns the time as a string in 12-hour format. The func. must return "09:15:00 AM" for the quarter past nine before noon (5p).

```
string Time::toString12(){
      string str = "";   int i=0;
      if(hour<10){  // saatin önüne 0
eklemek
        str = '0';
      }
      str += to_string(hour) + ":";
      if(min<10){   // dakika önüne 0
eklemek
        str += "0";
      }
      str += to_string(min) + ":";
      if(sec<10){   // saniye önüne 0
eklemek
        str += "0";
      }
      str += to_string(sec);
      if(ampm == 'a'){  // AM, PM kontrolü
        str += " AM";
      }
      else{     str += " PM";       }
      return str;
}
```

1.d) Saati 24 saat formatında string olarak geri döndüren `toString24()` fonksiyonunu yazınız. Akşam saat 9'u çeyrek geçe için "21:15:00" geri döndürmesini bekliyoruz - Write the function `toString24()`, which returns the time as a string in 24-hour format. This func. must return "21:15:00" for the quarter past nine in the afternoon (5p),

```
string Time::toString24(){
    string str = "";
    int i=0;
    if(ampm == 'p'){  // öğleden sonra ise +12
      hour += 12;
    }
    if(hour<10){   // saatin önüne 0 eklemek
      str = '0';
    }
    str += to_string(hour) + ":";
    if(min<10){   // dakika önüne 0 eklemek
      str += "0";
    }
    str += to_string(min) + ":";
    if(sec<10){  // saniye önüne 0 eklemek
      str += "0";
    }
    str += to_string(sec);
    return str;
}
```

1.e) + operatörünün overload işlemi, parametre dakika cinsinden bir değer olarak alınmalıdır → operator overloading for +, The overload of the operator + must be taken as a value in minutes (`t2 = t+20`) (10p)

```
Time Time::operator + (int _min){
  Time t1;
  t1.sec = this->sec;   // saniye aynen eşit
  t1.min = this->min + _min;  // dakika arttır
  if(t1.min>=60){   // taşma durumu
    t1.hour = this->hour + t1.min/60;
    t1.min = t1.min%60;
  }
  if(t1.hour >= 12){   // saatin taşma durumu
    t1.hour = t1.hour%12;
    if(t1.ampm == 'a'){  t1.ampm = 'p';  }
    else{  t1.ampm = 'a';    }
  }
  return t1;
}
```

2) Sizden bir **Date** sınıfı tanımlamanız istenmektedir. Bu sınıfın `yıl, ay, gün` şeklinde özellikleri bulunmaktadır. Bu değerlerin varsayılanı sırasıyla (1900,1,1) olmalıdır. Parametreli yapıcı fonksiyon kullanarak `yıl,ay,gün` değerleri setlenmelidir. - You are asked to define a **Date** class. This class has `year, month, day` properties. The default values of the properties must be (1900,1,1). `Year, month, day` values should be set by using parameterized constructor function.

2.a) Bu sınıfın parametreli, parametresiz yapıcı fonksiyonları, get set fonksiyonları ile b,c,d maddelerindeki fonksiyonları içerecek şekilde sınıfı tanımlayınız - Declare the default and parameterized constructors, getter and setter functions and the functions used in b,c,d. (5p).

```
class Date{
  public:
    Date(int _year=2000,int _month=1, int
_day=1);
    bool isLeap();
    string toString();
    Date operator + (const int _day);
  private:
    int year;
    int month;
    int day; };
```

2.b) Parametreli yapıcı fonksiyonu gerçekleyiniz - Write the parameterized constructor function code (5p).

```
Date::Date(int _year,int _month, int _day){
  this->year  = _year;
  this->month = _month;
  this->day   = _day;
}
```

2.c) Tarihin artık yıl olup olmadığını kontrol eden isLeap() fonksiyonu tanımlanmalıdır. (Vikipedi:Genel bir kural olarak, artık yıllar 4 rakamının katı olan yıllardır, ancak 100'ün katı olan yıllardan sadece 400'e kalansız olarak bölünebilenler artık yıldır) - write the function isLeap() that checks if the date is leap. (Wikipedia: As a general rule, every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400) (5p).

```
bool Date::isLeap(){
  if( (this->year%4==0 && this->year%100 != 0)
|| this->year%400 == 0)
    return true;
  return false;
}
```

2.d) Tarihi “01.01.2019” şeklinde string olarak geri döndüren fonksiyonu yazınız - Write the function that returns the date as string in “01.01.2019” format (5p).

```
string Date::toString(){
 string str="";
 if(this->day <10){
   str += "0";
 }
 str += to_string(this->day) + ".";
 if(this->month <10){
   str += "0";
 }
 str += to_string(this->month) + "." + to_string(this->year);
 return str;   }
```

2.e) + operatörünün overload işlemi, parametre gün cinsinden bir değer olarak alınmalıdır (artık yıl durumunu dikkate alınız) → operator overloading for +, The overload of the operator + must be taken as a value in days, (please consider the leap year status) (`d2 = d+20`) (10p)

```
Date Date::operator+(const int _day){
  Date d1;
  int d,m,y;
  d = this->day;
  m = this->month;
  y = this->year;
  int dArray[] =
{31,28,31,30,31,30,31,31,30,31,30,31};
  if(isLeap()) dArray[1] = 29;
  d += _day;
  if(d>dArray[m-1]){
    d = d%dArray[m-1];
    m++;
  }
  if(m>12){   m=1;y++;  }
  d1.year   = y;
  d1.month  = m;
  d1.day    = d;
  return d1;
}
```

3) Soru 1'de verilen Time sınıfı ve Soru 2'de verilen Date sınıfı kullanılarak DateTime sınıfı tanımlanıyor. Aşağıdaki kodda verilen boşlukları doldurunuz ve ana fonksiyondaki kod çalıştırıldığında ekran çıktısını yazınız - The DateTime class is defined by using the Date class in Question 1 and the Time class given in Question 2. Fill in the spaces given in the code below and write the screen output when the code in the main function is executed (5+2+3=10p).

```
class DateTime{
    Date date;  Time time;
  public:
    DateTime(int _year=1900,int _month=1, int
_day=1,int _hour=0, int _min=0, int _sec=0,int
_ampm='a'){
      Time t1(_hour,_min,_sec,_ampm);
      this->time = t1;
      Date d1(_year,_month,_day);
      this->date = d1;
    }
    string toString(){
return date.toString()+" "+time.toString24();
    }
    void setTime(Time _time)  {time = _time;}
    void setDate(Date _date)  {date = _date;}
};
int main() {
  Time t(3,59,0,'p');
  cout<<(t+20).toString24()<<endl;  // 16:19:00
  Date d(2000,12,20);
  cout<<(d+20).toString()<<endl;  // 09.01.2001
  DateTime dt;
  dt.setTime(Time(23,59,2)+40);
  dt.setDate(Date(2018,2,25)+20);
cout<<dt.toString()<<endl;//17.03.2018 00:39:02
return 0;  }
```

4) Ad ve haftalık çalışma saati, taban maaş ve maaş katsayısına sahip **Employee** (calişan) sınıfı ve bu sınıftan türetilen **Officer** (idari personel) ve **Lecturer** (akademik personel) sınıflarının gerçeklenmesi istenmektedir - You are required to implement an **Employee** class with name weekly working hours, base salary and salary coefficient properties. **Officer** and **Lecturer** classes must be derived from this class.

4.a) Bu sınıfları nesne yönelimli programlama ilkelerini de gözetecek şekilde tasarlayıp sınıflar için deklerasyonları içeren h dosyalarının içeriğini yazınız. Method olarak sadece parametreli ve parametresiz yapıcı fonksiyonlar ile haftalık çalışma saatini arttırmak için gerekli `incrementHour()` ve maaş hesaplaması yapan `getSalary()` fonk. nunu yazınız - write down the contents of the h files that contain the declarations for these classes. Consider the object oriented design principles. Please declare only the following methods: default and parameterized constructors, `incrementHour()` that increase the weekly working hours, `getSalary()` that returns the calculated salary in b. (10p).

```
// ----------  employee.h  ------------
class Employee {
  protected:
    string name;
    double sumOfHours;
    double baseSalary;
    double coefficient;
  public:
    Employee();
    Employee(string _name,double _sumOfHours,
double _baseSalary, double _coefficient);
    void incrementHour(double _hour=4);
    virtual double getSalary();
};
//------- ~ employee.h ------------------

// ------- officer.h ---------------
class Officer:public Employee{
  public:
    Officer(string _name,double _sumOfHours,
double _baseSalary, double _coefficient);
    virtual double getSalary();
};
// ------- ~officer.h ---------------

// ------- Lecturer.h ---------------
class Lecturer:public Employee{
  public:
    Lecturer(string _name,double _sumOfHours,
double _baseSalary, double _coefficient);
    virtual double getSalary();

};
// ------- ~ Lecturer.h ---------------
```

4.b) Bu sınıflarının yapıcı fonksiyonlarını, `incrementHour()` fonk. ve `getSalary()` fonksiyonunu gerçekleyiniz. Parametresiz yapıcı fonksiyonda özellikler 10 değerli varsayılan olarak alınmalıdır - write down the default and the parameterized constructors, `incrementHour()` and `getSalary()` functions (10p).

Maaş hesaplamak için - the calculating of the salary →
Employee → baseSalary + coefficient * sumOfHours + 500;
Officer → baseSalary + coefficient * sumOfHours + 1000;
Lecturer → baseSalary + coefficient * sumOfHours + 1500;

```
// ----- employee.cpp --------------
Employee::Employee(){
      name = "Unknown Name";
      sumOfHours = 0;
      baseSalary = 2020.0;
      coefficient = 10;
    }
Employee::Employee(string _name,double
_sumOfHours, double _baseSalary, double
_coefficient){
      name = _name;
      sumOfHours = _sumOfHours;
      baseSalary = _baseSalary;
      coefficient = _coefficient;
}
void Employee::incrementHour(double _hour){
      sumOfHours += _hour;
}
double Employee::getSalary(){
  return baseSalary + coefficient * sumOfHours
+ 500;
} // ----- ~ employee.cpp --------------

// ------- officer.cpp ---------------
Officer::Officer(string _name,double
_sumOfHours, double _baseSalary, double
_coefficient):Employee(_name,_sumOfHours,_baseS
alary,_coefficient){
  ;
}
double Officer::getSalary(){
  return baseSalary + coefficient * sumOfHours
+ 1000.0;
} // ------- ~ officer.cpp ---------------

// ------- Lecturer.cpp ---------------
Lecturer::Lecturer(string _name,double
_sumOfHours, double _baseSalary, double
_coefficient):Employee(_name,_sumOfHours,_baseS
alary,_coefficient){
  ;
}
double Lecturer::getSalary(){
  return baseSalary + coefficient * sumOfHours
+ 1500.0;
}  // ------- ~ Lecturer.cpp ---------------
```

4.c) Ana fonksiyon içerisinde aşağıdaki kod yazıldığında çalışanlara ödenen toplam maaş hesaplanarak ekrana yazılacaktır. Buna uygun `calculateTotalSalary()` fonk. gerçekleyiniz ve ekran çıktısını veriniz - When the following code is written in the main function, the output must be the total salary paid to employees. Please implement the `calculateTotalSalary()` function and give the screen output (8+2=10p).

```
double calculateTotalSalary(Employee *emp[],
int numOfEmp){

  double totalSalary = 0.0;
  for(int i=0;i<numOfEmp;i++){
    totalSalary += emp[i]->getSalary();
  }
  return totalSalary;

}

int main() {
  Employee *emp[3];
  Employee ali("Ali",40,1020.0,25);
  ali.incrementHour(8);
  Officer ayse("Ayse",40,1540.0,48);
  Lecturer veli("Veli",40,1860.0,72);
  emp[0] = &ali;
  emp[1] = &ayse;
  emp[2] = &veli;
  cout<<calculateTotalSalary(emp,3);
  return 0;
}
```

Ekran: 13420

Sınav Kuralları - Exam Rules

1. Bölüm sayfasında ilan edilen sınav kurallarına uymak zorunludur - It is compulsory to follow the rules of the examination announced on the web site.
2. Sorular 2 ve 3 numaralı program çıktıları ile ile ilişkilidir - Questions are associated with program outputs 2 and 3.

BAŞARILAR - GOOD LUCK