Objective:

- Revisiting character arrays and pointers.
- Basic Use of struct keyword.

Challenge-A *memmove* (5)

Write your own memmove function which we discussed in this week. But while achieving this goal, you are not allowed to use heap memory or any extra array.

```
void myMemMove(void * dest, const void * src, int byteCount)
```

Challenge-B *Time struct* (10 (2(1~4), 1(5~7), 1(8), 3(9), 3(10~12))

Implement the struct 'Time', which represents/store the time in 24-hour format.

Struct Members:

```
int hour;
int minute;
int second;
```

Your program should provide the implementation of following functions related to Time struct

```
1. void setTime ( Time &, int, int, int );   // set hour, minute, second
2. void setHour ( Time *, int );             // set hour
3. void setMinute ( Time *, int );           // set minute
4. void setSecond ( Time *, int );           // set second
5. int getHour ( Time );                     // return hour
6. int getMinute ( Time );                   // return minute
7. int getSecond ( Time );                   // return second
8. void printTwentyFourHourFormat ( Time );  // print universal time
9. void printTwelveHourFormat ( Time );      // print standard time
10. void incSec ( Time &, int = 1 );         // increment in the second: Smart Logic Needed
11. void incMin ( Time &, int = 1 );         // increment in the minute: Smart Logic Needed
12. void incHour ( Time &, int = 1 );        // increment in the hour: Smart Logic Needed
```

Note:

- You must make sure that all of the functions provided above are capable enough of ignoring any wrong/invalid inputs from the client. For Example, setting a negative value, setting hour value to 24 etc.
- No looping mechanism is allowed to implement functions from 10 to 12.

See below the sample runs to get further understanding:

Code	Console Output

`Time t = {10, 56, 12};`
`printTwentyFourHourFormat(t);`
`setHour(&t, 15);`
`printTwentyFourHourFormat(t);`
`setHour(&t, 25);`
`printTwentyFourHourFormat(t);`
	10:56:12 AM
	15:56:12 AM
	15:56:12 AM

`Time t = {15, 56, 12};`
`incSec(t,70);`
`printTwelveHourFormat(t);`
	3:57:22 PM

Challenge-C *PersonHealth struct* (8 (3, 2, 3))

While exercising, you can use heart-rate monitor to see that your heart rate stays within a safe range suggested by your trainers and doctors. According to AHA (American Heart Association), the formula for calculating your maximum heart rate in beats per minute is 220 minus your age in years. Your target heart rate is a range that is 50~85% of your maximum heart rate. See the table below for few examples:

Age	Target heart rate zone	Maximum heart rate
30	50 percent: 190 x 0.50 = 95 bpm	190 beats per minute

	85 percent: 190 x 0.85 = 162 bpm	
35	93 to 157 bpm	185 beats per minute
60	80 to 136 bpm	160 bpm

Knowing your heart rate during workout sessions can help know whether you are doing too much or not enough, the AHA says. When people exercise in their "target heart zone," they gain the most benefits and improve their heart's health. When your heart rate is in the target zone you know "you are pushing the muscle to get stronger," Bauman said.

Create a struct called '*PersonHealth*'. The struct attributes should include the person's first name, last name and date of birth (consist of day, month and year).
For each attribute provide set and get functions as we did in challenge-A.
In addition to above said things, you should provide the implementation of the following functions:

- `int getAge ( PersonHealth )`
  It calculates and returns the person's age in years by extracting date of birth from *PersonHealth* struct object and current date from computer system date.

But how to get current/system date/time?
CTime library will help us in this regard, which you may explore in detail at home as per your interest but for now I am pasting code which may give you required stuff for this lab at least.

```
time_t t = time(NULL);
tm curTime = * localtime(&t);
cout<<curTime.tm_mday<<'-'<<curTime.tm_mon + 1<<'-'<<curTime.tm_year + 1900;
```

A short explanation of the stuff used above:

- `time(NULL)` returns the time since 00:00:00 UTC, January 1, 1970 in seconds.
- `time_t` is an alias of integral data type capable of holding value returned by `time(NULL)`
- `localtime` function converts the received `time_t` object into calendar time. It actually returns an object of type tm struct whose attributes are as follows:

Member	Type	Meaning	Range
tm_sec	int	seconds after the minute	0-61*
tm_min	int	minutes after the hour	0-59
tm_hour	int	hours since midnight	0-23
tm_mday	int	day of the month	1-31
tm_mon	int	months since January	0-11
tm_year	int	years since 1900	
tm_wday	int	days since Sunday	0-6
tm_yday	int	days since January 1	0-365
tm_isdst	int	Daylight Saving Time flag	

- `int getMaximumHeartRate ( PersonHealth )`
  It returns the person's maximum heart rate.

- `TargetHeartRateZone getTargetHeartRateZone ( PersonHealth )`
  It returns the person's target heart rate zone.
  For this you will design another struct named '*TargetHeartRateZone*' which will have following attributes: *minHRate* and *maxHRate*.
  This function will create an object of *TargetHeartRateZone* and will initialize its attributes as per person age and returns the object.
  Note: the value in the examples of target heart rate range are rounded, so you also have to store the rounded values.

"Some of the best programming is done on paper, really. Putting it into the computer is just a minor detail."

-- Max Kanat-Alexander --