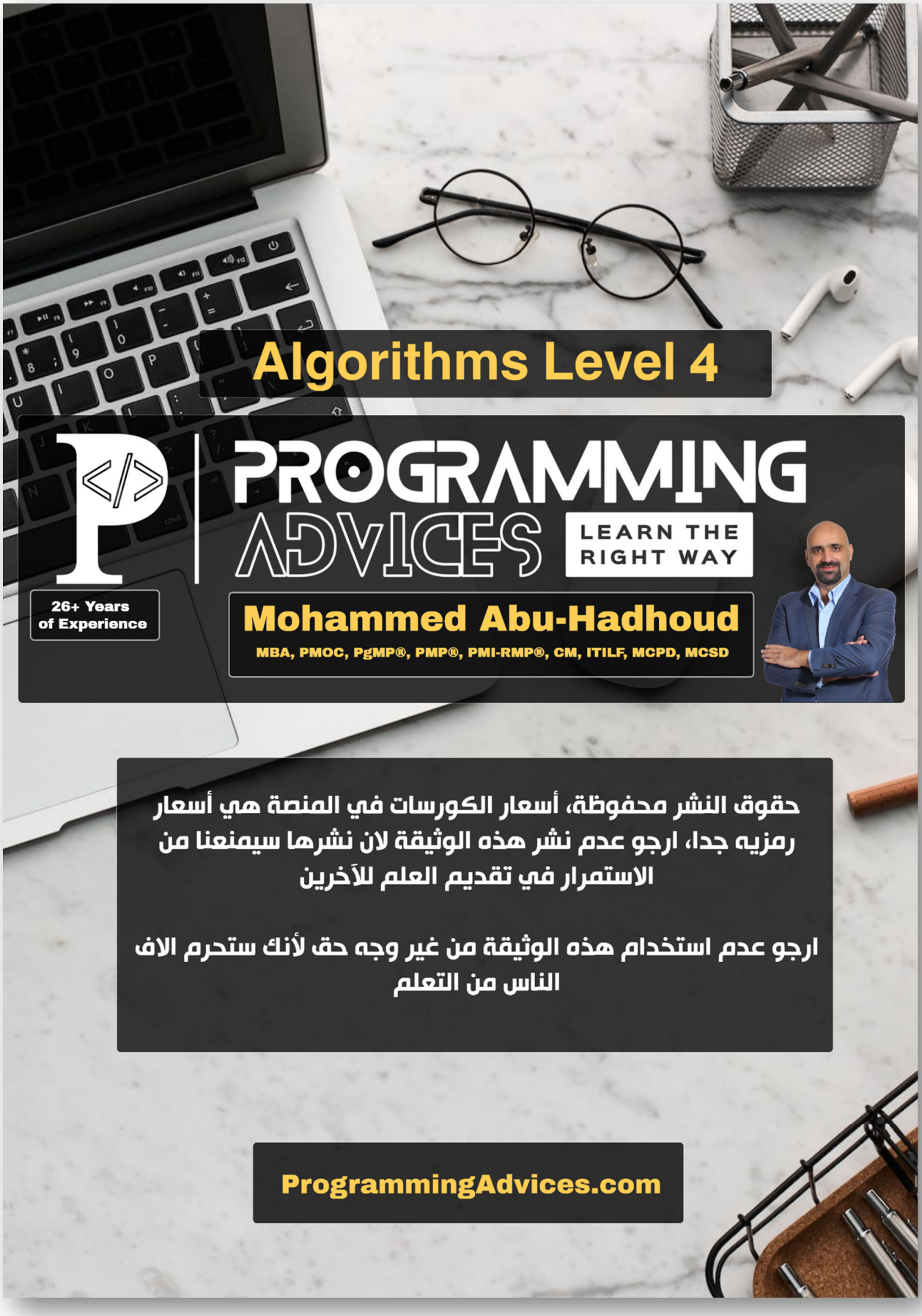
Algorithms Level 4
PROGRAMMING ADVICES
LEARN THE RIGHT WAY
26+ Years of Experience
Mohammed Abu-Hadhoud
MBA, PMOC, PgMP®, PMP®, PMI-RMP®, CM, ITILF, MCPD, MCSD
حقوق النشر محفوظة، أسعار الكورسات في المنصة هي أسعار رمزيه جدا، ارجو عدم نشر هذه الوثيقة لان نشرها سيمنعنا من الاستمرار في تقديم العلم للآخرين
ارجو عدم استخدام هذه الوثيقة من غير وجه حق لأنك ستحرم الاف الناس من التعلم
ProgrammingAdvices.com

```cpp
#include <iostream>
using namespace std;

struct stDate
{
    short Year;
    short Month;
    short Day;

};

struct stPeriod
{
    stDate StartDate;
    stDate EndDate;
};

bool IsDate1BeforeDate2(stDate Date1, stDate Date2)
{
    return  (Date1.Year < Date2.Year) ? true : ((Date1.Year ==
Date2.Year) ? (Date1.Month < Date2.Month ? true : (Date1.Month ==
Date2.Month ? Date1.Day < Date2.Day : false)) : false);
}

bool IsDate1EqualDate2(stDate Date1, stDate Date2)
{
    return  (Date1.Year == Date2.Year) ? ((Date1.Month ==
Date2.Month) ? ((Date1.Day == Date2.Day) ? true : false) : false)
: false;
}

bool IsDate1AfterDate2(stDate Date1, stDate Date2)
{
    return (!IsDate1BeforeDate2(Date1, Date2) &&
!IsDate1EqualDate2(Date1, Date2));

}

bool isLeapYear(short Year)
{
   return (Year % 4 == 0 && Year % 100 != 0) || (Year % 400 == 0);
}
```

```
short NumberOfDaysInAMonth(short Month, short Year)
{
    if (Month < 1 || Month>12)
        return  0;

    int days[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
    return (Month == 2) ? (isLeapYear(Year) ? 29 : 28) :
days[Month - 1];

}

bool IsLastDayInMonth(stDate Date)
{

    return (Date.Day == NumberOfDaysInAMonth(Date.Month,
Date.Year));

}

bool IsLastMonthInYear(short Month)
{
    return (Month == 12);
}

stDate IncreaseDateByOneDay(stDate Date)
{
    if (IsLastDayInMonth(Date))
    {
        if (IsLastMonthInYear(Date.Month))
        {
            Date.Month = 1;
            Date.Day = 1;
            Date.Year++;
        }
        else
        {
            Date.Day = 1;
            Date.Month++;
        }
    }
    else
    {
        Date.Day++;
    }

    return Date;
}
```

```
int GetDifferenceInDays(stDate Date1, stDate Date2, bool
IncludeEndDay = false)
{

    int Days = 0;
    while (IsDate1BeforeDate2(Date1, Date2))
    {
        Days++;
        Date1 = IncreaseDateByOneDay(Date1);
    }

    return IncludeEndDay ? ++Days : Days;
}


enum enDateCompare { Before = -1, Equal = 0, After = 1 };

enDateCompare CompareDates(stDate Date1, stDate Date2)
{

    if (IsDate1BeforeDate2(Date1, Date2))
        return enDateCompare::Before;

    if (IsDate1EqualDate2(Date1, Date2))
        return enDateCompare::Equal;

    /* if (IsDate1AfterDate2(Date1,Date2))
         return enDateCompare::After;*/

         //this is faster
    return enDateCompare::After;
}

int PeriodLengthInDays(stPeriod Period, bool IncludeEndDate =
false)
{
    return GetDifferenceInDays(Period.StartDate, Period.EndDate,
IncludeEndDate);
}
```

```
bool IsOverlapPeriods(stPeriod Period1, stPeriod Period2)
{

    if (
        CompareDates(Period2.EndDate, Period1.StartDate) ==
enDateCompare::Before
        ||
        CompareDates(Period2.StartDate, Period1.EndDate) ==
enDateCompare::After
        )
        return false;
    else
        return true;

}

bool isDateInPeriod(stDate Date, stPeriod Period)
{
    return !(CompareDates(Date, Period.StartDate) ==
enDateCompare::Before || CompareDates(Date, Period.EndDate) ==
enDateCompare::After);
}
```

```
int CountOverlapDays(stPeriod Period1, stPeriod Period2)
{
    int Period1Length = PeriodLengthInDays(Period1, true);
    int Period2Length = PeriodLengthInDays(Period2, true);
    int OverlapDays = 0;

    if (!IsOverlapPeriods(Period1, Period2))
        return 0;


    if (Period1Length < Period2Length)
    {
        while (IsDate1BeforeDate2(Period1.StartDate,
Period1.EndDate))
        {

            if (isDateInPeriod(Period1.StartDate, Period2))
                OverlapDays++;

            Period1.StartDate =
IncreaseDateByOneDay(Period1.StartDate);
        }
    }

    else
    {
        while (IsDate1BeforeDate2(Period2.StartDate,
Period2.EndDate))
        {

            if (isDateInPeriod(Period2.StartDate, Period1))
                OverlapDays++;

            Period2.StartDate =
IncreaseDateByOneDay(Period2.StartDate);
        }

    }

    return OverlapDays;

}
```

```
short ReadDay()
{
    short Day;
    cout << "\nPlease enter a Day? ";
    cin >> Day;
    return Day;
}

short ReadMonth()
{
    short Month;
    cout << "Please enter a Month? ";
    cin >> Month;
    return Month;
}

short ReadYear()
{
    short Year;
    cout << "Please enter a Year? ";
    cin >> Year;
    return Year;
}

stDate ReadFullDate()
{
    stDate Date;

    Date.Day = ReadDay();
    Date.Month = ReadMonth();
    Date.Year = ReadYear();

    return Date;
}

stPeriod ReadPeriod()
{
    stPeriod Period;
    cout << "\nEnter Start Date:\n";
    Period.StartDate = ReadFullDate();
    cout << "\nEnter End Date:\n";
    Period.EndDate = ReadFullDate();
    return Period;
}
```

```
int main()
{
    cout << "\nEnter Period 1 :";
    stPeriod Period1 = ReadPeriod();

    cout << "\nEnter Period 2 :";
    stPeriod Period2 = ReadPeriod();

    cout << "\nOverlap Days Count Is: " <<
CountOverlapDays(Period1, Period2);

    system("pause>0");
    return 0;
}
```