Date

در این سوال قرار است کلاس تاریخ را پیاده سازی کنیم.

مشخصا این کلاس شامل member variable های روز، ماه و سال خواهد بود (همگی به صورت int) و ممکن است نیاز به متغیر هایی داشته باشید که در بین تمامی کلاس هاتون یکسان باشه.

قرار است کلاس تاریخی که طراحی می کنیم نشان دهنده ی تاریخ های میلادی باشد. بنابراین حداقل سالی که مورد قبول می باشد 1973 و حداکثر آن 9999 می باشد همچنین روز و ماه نیز از 1 شروع می شوند.

نام ماه ها به ترتیب زیر می باشد.

```
{"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct",
```

نام روز ها نیز به ترتیب زیر می باشد.

```
{"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sat
```

و تعداد روز ها در ماه هم به صورت زیر می باشد.

```
{31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

البته باید حواسمان به تعداد روز ها در ماهی که منتهی به سال کبیسه می شود هم باشیم که در ادامه بهش می رسیم.

لطفا از اینجا به بعد را یک بار کامل بخونید. ممکن است توابعی داشته باشین که بتونین توی توابع دیگه صداشون بزنین :)

کلاستان شامل یک سازنده می باشد که با گرفتن روز، ماه و سال، member variable های کلاستون رو مقدار دهی می کند. (این اطمینان داده می شود که حتما ورودی های معتبری به سازنده داده شود)

کلاستان دارای member function های زیر می باشد:

- تابع SetDate که با گرفتن روز و ماه و سال در صورت معتبر بودنشان متغیر های داخل کلاس را مقدار دهی می کند.
- تابع SetYear که سال را مشخص می کند و در صورتی که min یا max سال را رعایت نکرده باشد پیامی مناسب به آن چاپ می کند.
- تابع GetYear
- تابع SetMonth که با گرفتن یک عدد صحیح ماه را مشخص می کند و در صورتی که ماه معتبری وارد نشده باشد پیامی مناسب به آن چاپ می کند.
- تابع GetMonth
- تابع SetDay که روز را مشخص می کند. این اطمینان به ما داده می شود که سال و ماه قبلا تنظیم (set) شده اند و با توجه به آن ها در صورتی که روز معتبری وارد نشده بود پیام مناسبی چاپ کند.
- تابع Print که تاریخ را چاپ می کند. (فرمت چاپ کردن در مثال زیر آورده شده است)
- تابع NextDay که تاریخ را یک روز جلو برده و آبجکت جاری را باز می گرداند.
- تابع PreviousDay که تاریخ را یک روز به عقب برده و آبجکت جاری باز می گرداند.
- تابع NextMonth
- تابع PreviousMonth
- تابع NextYear
- تابع PreviousYear

نکته: انتظار داریم در هنگام صدا زدن توابع NextDay تا PreviousYear هیچ گونه مخربی صدا زده نشود!!! همچنین باید حتما چک کنید با تغییر دادن روز، ماه یا سال همچنان داده هایتان معتبر باشد.

- تابع IsLeapYear که با گرفتن یک عدد صحیح به عنوان سال در صورتی که سال کبیسه باشد true و در غیر این صورت false بر میگرداند.

نکته: در تقویم میلادی، سالی کبیسه است که شرایط زیر را داشته باشد:

(مضربی از ۴۰۰ باشد) یا (مضربی از ۴ باشد و مضربی از ۱۰۰ نباشد.)

همچنین در صورتی که سال کبیسه باشد ماه دوم یعنی Feb به جای 28 روز 29 روز خواهد شد.

- تابع IsValidDate این تابع با گرفتن روز و ماه و سال می گوید که آیا این تاریخ معتبر است یا خیر. (true or false)
- تابع GetDayOfWeek این تابع با گرفتن روز و ماه و سال به ما روز آن هفته را می گوید (در صورتی که یکشنبه باشد مقدار 0 و تا شنبه که مقدار 6 می گیرد) برای پیدا کردن روز می توانید از الگوریتم های موجود در این لینک ویکیپدیا استفاده کنید.
- تابع PrintDayOfWeek که با گرفتن روز و ماه و سال روز آن هفته را چاپ می کند.

مثال

ورودی نمونه

```
Date date1(2020, 1, 1);
date1.Print();
date1.NextDay().Print();
date1.Print();

date1.SetDate(2021, 1, 31);
date1.Print();
date1.NextDay().Print();

date1.SetDate(2020, 2, 28);
date1.Print();
date1.NextDay().Print();

date1.SetDate(2020, 12, 31);
date1.Print();
date1.NextDay().Print();

date1.SetYear(1600);
date1.Print();
date1.SetDay(31);
date1.Print();
date1.NextMonth().NextMonth().NextMonth();
date1.Print();
date1.SetDay(31);
date1.Print();
```

```
if (Date::IsValidDate(2021, 2, 29))
{
    cout << "Date is valid" << endl;
}
else
{
    cout << "Date is not valid" << endl;
}

if (Date::IsLeapYear(2000))
{
    cout << "2000 is leap year" << endl;
}
else
{
    cout << "2000 is not leap year" << endl;
}

if (Date::IsValidDate(2021, 1, 1))
{
    Date::PrintDayOfWeek(2021, 1, 1);
}

Date* date4 = nullptr;
date4 = new Date(9999, 12, 30);
date4->NextDay().Print();
date4->NextDay();
date4->Print();
delete date4;

Date date5(2021, 1, 1);
date5.PreviousDay().Print();

Date date6(2021, 3, 31);
date6.NextMonth().Print();

Date date7(2020, 3, 31);
date7.PreviousMonth().Print();

Date date8(2020, 2, 29);
```

```
    date8.NextYear().Print();

    Date date9(2020, 2, 29);
    date9.PreviousYear().Print();
```

خروجی نمونه

```
Wednesday, 1 Jan 2020
Thursday, 2 Jan 2020
Thursday, 2 Jan 2020
Sunday, 31 Jan 2021
Monday, 1 Feb 2021
Friday, 28 Feb 2020
Saturday, 29 Feb 2020
Thursday, 31 Dec 2020
Friday, 1 Jan 2021
Error: Invalid year (1753-9999)!
Friday, 1 Jan 2021
Sunday, 31 Jan 2021
Wednesday, 28 Apr 2021
Error: Invalid day! your input is: 31 but last day of month is : 30
Wednesday, 28 Apr 2021
Date is not valid
2000 is leap year
Friday, 1 Jan 2021
Friday, 31 Dec 9999
Error: Next day is out of range!
Friday, 31 Dec 9999
Thursday, 31 Dec 2020
Friday, 30 Apr 2021
Saturday, 29 Feb 2020
Sunday, 28 Feb 2021
Thursday, 28 Feb 2019
```