بسم الله الرحمن الرحيم

# Problem 1 : Number To Text

Write a program to read a number and print the Text of that number

```
Please enter Number ?
546780834

Five Hundreds Forty Six Millions
Seven Hundreds Eighty
Thousands Eight Hundreds
Thirty Four
```

```cpp
#include <iostream>
#include <string>

using namespace std;

// Problem #1

string NumberToText(int Number)
{
	if (Number == 0)
	{
		return "";
	}

	if(Number >= 1 && Number <= 19)
	{
		string arr[] = { "", "One" , "Two" ,
"Three" , "Four" , "Five" , "Six" , "Seven" ,
"Eight" , "Nine" , "Ten" , "Eleven" , "Twelve" ,
 "Thirteen" , "Fourteen" , "Fifteen" , "Sixteen" ,
"Seventeen" , "Nineteen" };

		return arr[Number] + " ";
	}

	if (Number >= 20 && Number <= 99)
	{
		string arr[] = { "" , "" , "Twenty" , "Thirty" , "Forty" , "Fifty" ,
		                   "Sixty", "Seventy" , "Eighty" , "Ninety"};

		return arr[Number / 10] + " " + NumberToText(Number % 10); [1]
	}

	if (Number >= 100 && Number <= 199)
	{
		return "One Hundred " + NumberToText(Number % 100);
	}

	if (Number >= 200 && Number <= 999)
	{
		return NumberToText(Number / 100) + "Hundreds " +
NumberToText(Number % 100);
	}

	if (Number >= 1000 && Number <= 1999)
	{
		return "One Thousand " + NumberToText(Number % 1000);
	}
	if (Number >= 2000 && Number <= 999999)
	{
		return NumberToText(Number / 1000) + "Thousands " +
NumberToText(Number % 1000);
	}
	if (Number >= 1000000 && Number <= 1999999)
	{
		return  "One Million " + NumberToText(Number % 1000000);
	}

	if (Number >= 2000000 && Number <= 999999999)
	{
		return   NumberToText(Number / 1000000) + "Millions " + 
NumberToText(Number % 1000000);
	}

	if (Number >= 1000000000 && Number <= 1999999999)
	{
		return  "One Billion " + NumberToText(Number % 1000000000);
	}
	else
	{
		return   NumberToText(Number / 1000000000) + "Billions " + 
NumberToText(Number % 1000000000);
	}

}

int ReadNumber()
{
	int Num = 0;

	cout << "\nPlease enter Number ? ";
	cin >> Num;

	return Num;
}

int main()
{
	int Number = ReadNumber();

	cout << NumberToText(Number);

	system("pause>0");

	return 0;
}
```

#Problem 2 : Leap Year

```cpp
#include <iostream>
#include <string>

using namespace std;
// Problem #2

bool IsLeapYear(short Year)
{
	// leap year if perfectly divisible by 400
	if (Year % 400 == 0)
	{
		return true;
	}
	// not a leap year if divisible by 100
	// but not divisible by 400
	else if (Year % 100 == 0)
	{
		return false;
	}
	// leap year if not divisible by 100
	// but divisible by 4
	else if (Year % 4 == 0)
	{
		return true;
	}
	// all other years are not leap years
	else
	{
		return false;
	}
}

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check? ";
	cin >> Year;

	return Year;
}

int main()
{
	// Problem #2

	short Year = ReadYear();

	if (IsLeapYear(Year))
		cout << "Yes , Year [" << Year << "] is a leap year. \n";
	else
		cout << "No , Year [" << Year << "] is NOT a leap year. \n";


	system("pause>0");

	return 0;
}
```

Write a program to check if Year is a Leap Year or NOT

```
Please enter a year to check ?
1900

No , Year [1900] is NOT a leap
.year



Please enter a year to check ?
2000

.Yes , Year [2000] is a leap year
```

#Problem 3 : Leap Year (One Line Of Code)

```
#include <iostream>
#include <string>

using namespace std;

// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4 AND not 
divisible by 100
	// OR if year is divisible by 400
	// then it is a leap year

	return (Year % 4 == 0 && Year % 100 != 0) || 
(Year % 400 == 0);
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

int main()
{
	// Problem #3

	short Year = ReadYear();

	if (IsLeapYear(Year))
		cout << "Yes , Year [" << Year << "] is a leap year. \n";
	else
		cout << "No , Year [" << Year << "] is NOT a leap year. \n";


	system("pause>0");

	return 0;
}
```

Write a program to check if Year is a Leap Year or NOT

```
Please enter a year to check ?
1900

No , Year [1900] is NOT a leap
.year



Please enter a year to check ?
2000

.Yes , Year [2000] is a leap year
```

#Problem 4 : Number of Days – Hours – Minutes – Seconds in a Year

```
#include <iostream>
#include <string>

using namespace std;

// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4 AND not 
divisible by 100
	// OR if year is divisible by 400
	// then it is a leap year

	return (Year % 4 == 0 && Year % 100 != 0) || 
(Year % 400 == 0);
}

// Problem #4

short NumberOfDaysInAYear(short Year)
{
	return IsLeapYear(Year) ? 366 : 365;
}
short NumberOfHoursInAYear(short Year)
{
	return NumberOfDaysInAYear(Year) * 24;
}
int NumberOfMinutesInAYear(short Year)
{
	return NumberOfHoursInAYear(Year) * 60;
}
int NumberOfSecondsInAYear(short Year)
{
	return NumberOfMinutesInAYear(Year) * 60;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

Write a program to print Number Of : Days / Hours / Minutes / Seconds in a certain Year

```
Please enter a year to check ?
2000

Number of Days in Year
[2000] is 366

Number of Hours in Year
[2000] is 8784

Number of Minutes in Year
[2000] is 527040

Number of Seconds in Year
[2000] is 31622400


Please enter a year to check ?
1900

Number of Days in Year
[1900] is 365

Number of Hours in Year
[1900] is 8760

Number of Minutes in Year
[1900] is 525600

Number of Seconds in Year
[1900] is 31536000
```

```cpp
int main()
{

       // Problem #4

       short Year = ReadYear();

       cout << "\nNumber of Days in Year    [" << Year << "] is "
              << NumberOfDaysInAYear(Year);

       cout << "\nNumber of Hours in Year   [" << Year << "] is "
              << NumberOfHoursInAYear(Year);

       cout << "\nNumber of Minutes in Year [" << Year << "] is "
              << NumberOfMinutesInAYear(Year);

       cout << "\nNumber of Seconds in Year [" << Year << "] is "
              << NumberOfSecondsInAYear(Year);


       system("pause>0");

       return 0;
}
```

#Problem 5 : Number of Days – Hours – Minutes – Seconds in a Month

```
#include <iostream>
#include <string>

using namespace std;

// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4 AND not
divisible by 100
	// OR if year is divisible by 400
	// then it is a leap year

	return (Year % 4 == 0 && Year % 100 != 0) ||
(Year % 400 == 0);
}

// Problem #5

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

	if (Month == 2)
	{
		return  IsLeapYear(Year) ? 29 : 28;
	}

	short arr31Days[7] = { 1,3,5,7,8,10,12 };

	for (short i = 1; i <= 7; i++)
	{
		if (arr31Days[i - 1] == Month)
			return 31;
	}
	//if you reach here then its 30 days.
	return  30;
}
short NumberOfHoursInAMonth(short Month , short
Year)
{
	return NumberOfDaysInAMonth(Month , Year) *
24;
}
int NumberOfMinutesInAMonth(short Month , short
Year)
{
	return NumberOfHoursInAMonth(Month , Year) *
60;
}
int NumberOfSecondsInAMonth(short Month , short
Year)
{
	return NumberOfMinutesInAMonth(Month , Year)
* 60;
}
```

Write a program to print Number Of : Days / Hours / Minutes / Seconds in a certain Month

```
Please enter a year to check ?
1999

Please enter a Month to check
? 12

Number of Days in Month
[12] is 31

Number of Hours in Month
[12] is 744

Number of Minutes in Month
[12] is 44640

Number of Seconds in Month
[12] is 2678400


Please enter a year to check ?
2000

Please enter a Month to check
? 2

Number of Days in Month    [2]
is 29

Number of Hours in Month   [2]
is 696

Number of Minutes in Month
[2] is 41760

Number of Seconds in Month
[2] is 2505600
```

```cpp
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

int main()
{
	// Problem #5

	short Year = ReadYear();
	short Month = ReadMonth();

	cout << "\nNumber of Days in Month    [" << Month << "] is "
		<< NumberOfDaysInAMonth(Month, Year);

	cout << "\nNumber of Hours in Month   [" << Month << "] is "
		<< NumberOfHoursInAMonth(Month, Year);

	cout << "\nNumber of Minutes in Month [" << Month << "] is "
		<< NumberOfMinutesInAMonth(Month, Year);

	cout << "\nNumber of Seconds in Month [" << Month << "] is "
		<< NumberOfSecondsInAMonth(Month, Year);


	system("pause>0");

	return 0;
}
```

#Problem 6 : Number of Days in a Month Short Logic

```cpp
#include <iostream>
#include <string>

using namespace std;

// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4 AND not
divisible by 100
	// OR if year is divisible by 400
	// then it is a leap year

	return (Year % 4 == 0 && Year % 100 != 0) ||
(Year % 400 == 0);
}

// Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) [2] : NumberOfDays[Month - 1];

}

short NumberOfHoursInAMonth(short Month , short
Year)
{
	return NumberOfDaysInAMonth(Month , Year) *
24;
}

int NumberOfMinutesInAMonth(short Month , short
Year)
{
	return NumberOfHoursInAMonth(Month , Year) *
60;
}

int NumberOfSecondsInAMonth(short Month , short
Year)
{
	return NumberOfMinutesInAMonth(Month , Year)
* 60;
}
```

Write a program to print Number Of : Days in a certain Month

```
Please enter a year to check ?
1999

Please enter a Month to check
? 12

Number of Days in Month
[12] is 31

Number of Hours in Month
[12] is 744

Number of Minutes in Month
[12] is 44640

Number of Seconds in Month
[12] is 2678400


Please enter a year to check ?
2000

Please enter a Month to check
? 2

Number of Days in Month    [2]
is 29

Number of Hours in Month   [2]
is 696

Number of Minutes in Month
[2] is 41760

Number of Seconds in Month
[2] is 2505600
```

Lesson #09 : Ternary Operator: Short Hand If [2]

```
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

int main()
{
	// Problem #6

	short Year = ReadYear();
	short Month = ReadMonth();

	cout << "\nNumber of Days in Month    [" << Month << "] is "
		<< NumberOfDaysInAMonth(Month, Year);

	cout << "\nNumber of Hours in Month   [" << Month << "] is "
		<< NumberOfHoursInAMonth(Month, Year);

	cout << "\nNumber of Minutes in Month [" << Month << "] is "
		<< NumberOfMinutesInAMonth(Month, Year);

	cout << "\nNumber of Seconds in Month [" << Month << "] is "
		<< NumberOfSecondsInAMonth(Month, Year);


	system("pause>0");

	return 0;
}
```

#Problem 7 : Day Name

Write a program to read a date , and print the Day Name of Week

```
Please enter a year to check ?
2023


Please enter a Month to check
? 8


Please enter a Day to check ?
12


Date        : 12/8/2023

Day Order   : 6

Day Name    : Sat
```

```cpp
#include <iostream>
#include <string>

using namespace std;

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

short DayOfWeekOrder(short Day, short Month, short Year)
{
	short a, y, m;

	a = (14 - Month) / 12;
	y = Year - a;
	m = Month + (12 * a) - 2;

	// Gregorian:
	//0:sun, 1:Mon, 2:Tue...etc.
	return (Day + y + (y / 4) - (y / 100) + (y / 400) + ((31 * m) / 12)) % 7;
}
```

```cpp
string DayShortName(short DayOfWeekOrder)
{
	string arrDayNames[7] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder];
}


int main()
{
	// Problem #7

	short Year = ReadYear();
	short Month = ReadMonth();
	short Day = ReadDay();

	short DayOrder = DayOfWeekOrder(Day, Month, Year);

	cout << "\n\nDate        : " << Day << "/" << Month << "/" << Year << 
endl;
	cout << "Day Order   : " << DayOrder << endl;
	cout << "Day Name   : " << DayShortName(DayOrder) << endl;


	system("pause>0");

	return 0;
}
```

#Problem 8 : Month Calendar

Write a program to print Month Calendar

```
Please enter a year to check ? 2023

Please enter a Month to check ? 8

_______________________Aug___________________

Sun  Mon  Tue  Wed  Thu  Fri  Sat

0     0    1    2    3    4    5

6    7    8    9    10   11   12

13   14   15   16   17   18   19

20   21    22   23   24   25   26

27   28   29   30   31

_____________________________________________
```

```cpp
#include <iostream>
#include <string>

using namespace std;
// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4
AND not divisible by 100
	// OR if year is divisible by
400
	// then it is a leap year

	return (Year % 4 == 0 && Year
% 100 != 0) || (Year % 400 == 0);
}

// Problem #7

short DayOfWeekOrder(short Day,
short Month, short Year)
{
	short a, y, m;

	a = (14 - Month) / 12;
	y = Year - a;
	m = Month + (12 * a) - 2;

	// Gregorian:
	//0:sun, 1:Mon, 2:Tue...etc.
	return (Day + y + (y / 4) -
(y / 100) + (y / 400) + ((31 * m) /
12)) % 7;
}

string DayShortName(short
DayOfWeekOrder)
{
	string arrDayNames[7] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder];
}

// Problem #6

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

	int NumberOfDays[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month -
1];

}
```

```
// Problem #8

string MonthShortName(short MonthNumber)
{
	string Months[12] =
	{
		"Jan", "Feb", "Mar",
		"Apr", "May", "Jun",
		"Jul", "Aug", "Sep",
		"Oct", "Nov", "Dec"
	};
	return (Months[MonthNumber - 1]);
}

void PrintMonthCalendar(short Month, short Year)
{
	int NumberOfDays;

	// Index of the day from 0 to 6
	int current = DayOfWeekOrder(1, Month, Year);

	NumberOfDays = NumberOfDaysInAMonth(Month, Year);

	// Print the current month name
	printf("\n _______________%s__________________\n\n",
		MonthShortName(Month).c_str());[3]

	// Print the columns
	printf("  Sun  Mon  Tue  Wed  Thu  Fri  Sat\n");

	// Print appropriate spaces
	int i;
	for (i = 0; i < current; i++)
		printf("     ");

	for (int j = 1; j <= NumberOfDays; j++)
	{
		printf("%5d", j);

		if (++i == 7)
		{
			i = 0;
			printf("\n");
		}
	}
	printf("\n ____________________________________\n");
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}
```

[3] Lesson #25 - String And Char Format (Printf)

```
// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}


int main()
{
	// Problem #8

	short Year = ReadYear();
	short Month = ReadMonth();

	PrintMonthCalendar(Month, Year);

	system("pause>0");

	return 0;
}
```

#Problem 9 : Year Calendar

```
Write a program to print Year Calendar

Please enter a year to check ? 2023

_______________________________________

            Calendar – 2023

_______________________________________

_____________________Jan_________________

Sat  Fri  Thu  Wed  Tue Mon Sun

7    6    5    4    3    2    1

14   13   12   11   10   9    8

21   20   19   18   17   16   15

28   27   26   25   24   23   22

31   30   29

__________________________________________



_____________________Feb_________________
```

```cpp
#include <iostream>
#include <string>

using namespace std;
// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4
AND not divisible by 100
	// OR if year is divisible by
400
	// then it is a leap year

	return (Year % 4 == 0 && Year
% 100 != 0) || (Year % 400 == 0);
}

// Problem #7

short DayOfWeekOrder(short Day,
short Month, short Year)
{
	short a, y, m;

	a = (14 - Month) / 12;
	y = Year - a;
	m = Month + (12 * a) - 2;

	// Gregorian:
	//0:sun, 1:Mon, 2:Tue...etc.
	return (Day + y + (y / 4) -
(y / 100) + (y / 400) + ((31 * m) /
12)) % 7;
}

string DayShortName(short
DayOfWeekOrder)
{
	string arrDayNames[7] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder];
}

// Problem #6

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

	int NumberOfDays[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month -
1];

}
```

```
// Problem #8

string MonthShortName(short MonthNumber)
{
	string Months[12] =
	{
		"Jan", "Feb", "Mar",
		"Apr", "May", "Jun",
		"Jul", "Aug", "Sep",
		"Oct", "Nov", "Dec"
	};
	return (Months[MonthNumber - 1]);
}

void PrintMonthCalendar(short Month, short Year)
{
	int NumberOfDays;

	// Index of the day from 0 to 6
	int current = DayOfWeekOrder(1, Month, Year);

	NumberOfDays = NumberOfDaysInAMonth(Month, Year);

	// Print the current month name
	printf("\n _______________%s__________________\n\n",
		MonthShortName(Month).c_str());

	// Print the columns
	printf("  Sun  Mon  Tue  Wed  Thu  Fri  Sat\n");

	// Print appropriate spaces
	int i;
	for (i = 0; i < current; i++)
		printf("     ");

	for (int j = 1; j <= NumberOfDays; j++)
	{
		printf("%5d", j);

		if (++i == 7)
		{
			i = 0;
			printf("\n");
		}
	}
	printf("\n ____________________________________\n");
}

// Problem #9

void PrintYearCalendar( short Year)
{
	printf("\n  _________________________________\n\n");
	printf("           Calendar - %d\n", Year);
	printf("  _________________________________\n");

	for (short i = 1; i <= 12; i++)
	{
		PrintMonthCalendar(i, Year);
	}
}
```

```cpp
// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}



int main()
{
	// Problem #9

	PrintYearCalendar(ReadYear());


	system("pause>0");

	return 0;
}
```

#Problem 10 : Days from the beginning of Year

Write a program to print Total Days from the beginning of Year

```
Please enter a Day to check ?
13

Please enter a Month to check
? 8

Please enter a year to check ?
2023



Number of Days from
beginning of the Year Is : 225
```

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

// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4 AND not
divisible by 100
	// OR if year is divisible by 400
	// then it is a leap year

	return (Year % 4 == 0 && Year % 100 != 0) ||
(Year % 400 == 0);
}

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

    return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month - 1];

}

// Problem #10

short NumberOfDaysFromTheBeginingOfTheYear(short Day, short Month, short Year)
{
	short TotalDays = 0;

	for (int i = 1 ; i <= Month - 1 ; i++)
	{
		TotalDays += NumberOfDaysInAMonth(i, Year);
	}
	TotalDays += Day;

	return TotalDays;
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}
```

```cpp
// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

int main()
{
	// Problem #10

	short Day = ReadDay();
	short Month = ReadMonth();
	short Year = ReadYear();

	cout << "\n\nNumber of Days from beginning of the Year Is : "
		<< NumberOfDaysFromTheBeginingOfTheYear(Day, Month, Year);


	system("pause>0");

	return 0;
}
```

#Problem 11 : Date from Day Order In a Year

Write a program to print Total Days from the beginning of Year , Then Tack the Total Days and convert them back to data

```
Please enter a Day to check ?
13

Please enter a Month to check
? 8

Please enter a year to check ?
2023



Number of Days from
beginning of the Year Is : 225



Date for [225] is: 13/8/2023
```

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

// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4 AND not
divisible by 100
	// OR if year is divisible by 400
	// then it is a leap year

	return (Year % 4 == 0 && Year % 100 != 0) ||
(Year % 400 == 0);
}

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

    return (Month == 2) ? (IsLeapYear(Year) ? 29 :
28) : NumberOfDays[Month - 1];

}

// Problem #10

short NumberOfDaysFromTheBeginingOfTheYear(short Day, short Month, short Year)
{
	short TotalDays = 0;

	for (int i = 1 ; i <= Month - 1 ; i++)
	{
		TotalDays += NumberOfDaysInAMonth(i, Year);
	}
	TotalDays += Day;

	return TotalDays;
}

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};
```

```
sDate GetDateFromDayOrderInYear(short DateOrderInYear, short Year)
{
	sDate Date;

	short RemainingDays = DateOrderInYear;
	short MonthDays = 0;
	Date.Year = Year;

	Date.Month = 1;

	while (true)
	{
		MonthDays = NumberOfDaysInAMonth(Date.Month, Year);

		if (RemainingDays > MonthDays)
		{
			RemainingDays -= MonthDays;
			Date.Month++;
		}
		else
		{
			Date.Day = RemainingDays;
			break;
		}
	}
	return Date;
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

```
int main()
{
	// Problem #11

	short Day = ReadDay();
	short Month = ReadMonth();
	short Year = ReadYear();

	short DaysOrderInYear = NumberOfDaysFromTheBeginingOfTheYear(Day, Month, 
Year);

	cout << "\n\nNumber of Days from beginning of the Year Is : "
		<< DaysOrderInYear << endl;

	sDate Date;
	Date = GetDateFromDayOrderInYear(DaysOrderInYear, Year);

	cout << "\nDate for [" << DaysOrderInYear << "] is: ";
	cout << Date.Day << "/" << Date.Month << "/" << Date.Year;

	system("pause>0");

	return 0;
}
```

#Problem 12 : Add Days to Date

Write a program to read how many days to add to it , print the results on screen

```
Please enter a Day to check ? 14

Please enter a Month to check ? 8

Please enter a year to check ? 2023

How many days to add? 2500


Date after adding [2500] days is: 18/6/2030
```

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

bool IsLeapYear(short Year)
{
	// if year is divisible by 4 AND not
divisible by 100
	// OR if year is divisible by 400
	// then it is a leap year

	return (Year % 4 == 0 && Year % 100 != 0) ||
(Year % 400 == 0);
}

//Problem #6

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

	int NumberOfDays[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

    return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month - 1];

}

// Problem #10

short NumberOfDaysFromTheBeginingOfTheYear(short Day, short Month, short Year)
{
	short TotalDays = 0;

	for (int i = 1 ; i <= Month - 1 ; i++)
	{
		TotalDays += NumberOfDaysInAMonth(i, Year);
	}
	TotalDays += Day;

	return TotalDays;
}
```

```
// Problem #12

sDate DateAddDays(short Days, sDate Date)
{
	short RemainingDays = Days +
		NumberOfDaysFromTheBeginingOfTheYear(Date.Day, Date.Month,
			Date.Year);
	short MonthDays = 0;
	Date.Month = 1;
	while (true)
	{
		MonthDays = NumberOfDaysInAMonth(Date.Month, Date.Year);
		if (RemainingDays > MonthDays)
		{
			RemainingDays -= MonthDays;
			Date.Month++;
			if (Date.Month > 12)
			{
				Date.Month = 1;
				Date.Year++;
			}
		}
		else
		{
			Date.Day = RemainingDays;
			break;
		}
	}
	return Date;
}

// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

```
sDate ReadFullDate()
{
	sDate Date;
	Date.Day = ReadDay();
	Date.Month = ReadMonth();
	Date.Year = ReadYear();
	return Date;
}

short ReadDaysToAdd()
{
	short Days;
	cout << "\nHow many days to add? ";
	cin >> Days;
	return Days;
}


int main()
{
	// Problem #12

	sDate Date = ReadFullDate();
	short Days = ReadDaysToAdd();

	Date = DateAddDays(Days, Date);

	cout << "\nDate after adding [" << Days << "] days is: ";
	cout << Date.Day << "/" << Date.Month << "/" << Date.Year;



	system("pause>0");

	return 0;
}
```

#Problem 13 : Date1 Less Than Date2

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

Write a program to read Date1 , Date2 and check if Date1 Less than Date2

```
Please enter a Day to check ?
15

Please enter a Month to check
? 8

Please enter a year to check ?
2023


Please enter a Day to check ?
15

Please enter a Month to check
? 9

Please enter a year to check ?
2023


.Yes, Date1 is Less than Date2
```

```
sDate ReadFullDate()
{
	sDate Date;
	Date.Day = ReadDay();
	Date.Month = ReadMonth();
	Date.Year = ReadYear();
	return Date;
}

short ReadDaysToAdd()
{
	short Days;
	cout << "\nHow many days to add? ";
	cin >> Days;
	return Days;
}

int main()
{
	// Problem #13

	sDate Date1 = ReadFullDate();
	cout << "\n\n";
	sDate Date2 = ReadFullDate();

	if (IsDate1BeforeDate2(Date1, Date2))
		cout << "\nYes, Date1 is Less than Date2.";

	else
		cout << "\nNo, Date1 is NOT Less than Date2.";

	system("pause>0");

	return 0;
}
```

#Problem 14 : Date1 Equals To Date2

Write a program to read Date1 , Date2 and check if Date1 Equals to Date2

```
Please enter a Day to check ?
15

Please enter a Month to check
? 8

Please enter a year to check ?
2023


Please enter a Day to check ?
15

Please enter a Month to check
? 8

Please enter a year to check ?
2023


.Yes, Date1 is Equal to Date2
```

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #14

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

// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ?
";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

```
sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

short ReadDaysToAdd()
{
	short Days;
	cout << "\nHow many days to add? ";
	cin >> Days;
	return Days;
}

int main()
{
	// Problem #14

	sDate Date1 = ReadFullDate();
	cout << "\n\n";
	sDate Date2 = ReadFullDate();

	if (IsDate1EqualDate2(Date1, Date2))
		cout << "\nYes, Date1 is Equal to Date2.";

	else
		cout << "\nNo, Date1 is NOT Equal to Date2.";



	system("pause>0");

	return 0;
}
```

#Problem 15 : Last Day , Last Month

Write a program to read and check

if it is last Day in Month if it is last Month in Year

```
Please enter a Day to check ? 31

Please enter a Month to check ? 8

Please enter a year to check ? 2023


.Yes, Day is Last In Month

.No, Month is NOT Last In Year
```

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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 NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) : NumberOfDays[Month - 1];

}

// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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


// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}
```

```
// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

int main()
{
	// Problem #15

	sDate Date = ReadFullDate();

	if (IsLastDayInMonth(Date))
		cout << "\nYes, Day is Last In Month.";

	else
		cout << "\nNo, Day is NOT Last In Month.";


	if (IsLastMonthInYear(Date.Month))
		cout << "\nYes, Month is Last In Year.";

	else
		cout << "\nNo, Month is NOT Last In Year.";


	system("pause>0");

	return 0;
}
```

#Problem 16 : Increase Date By One Day

Write a program to read Date and make function to Increase by one Day

```
Please enter a Day to check ? 31

Please enter a Month to check ? 12

Please enter a year to check ? 2023


Date after adding one Day is : 1/1/2024
```

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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 NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month -
1];

}

// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

```
// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

```cpp
// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

int main()
{
	// Problem #16

	sDate Date = ReadFullDate();

	Date = IncreaseDateByOneDay(Date);


	cout << "\nDate after adding one Day is : "
	<< Date.Day << "/" << Date.Month << "/" << Date.Year;


	system("pause>0");

	return 0;
}
```

#Problem 17 : Diff In Days

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) : NumberOfDays[Month - 1];

}
// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

Write a program to read Date1 , Date2 and make function to Calculate the Difference in Days

NOTS :Date1 should be less than Date2

```
Please enter a Day to check ? 1

Please enter a Month to check ? 1

Please enter a year to check ? 2023


Please enter a Day to check ? 16

Please enter a Month to check ? 8

Please enter a year to check ? 2023


.Difference is: 227 Day(s)

Difference (Including End Day) .is: 228 Day(s)
```

```cpp
// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #17

int GetDifferenceInDays(sDate Date1 , sDate Date2 , bool IncludeEndDay= false)
{
	int Days = 0;

	while (IsDate1BeforeDate2(Date1, Date2))
	{
		Days++;
		Date1 = IncreaseDateByOneDay(Date1);
	}
	return IncludeEndDay ? ++Days : Days;
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}
```

```
// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}



// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

int main()
{
	// Problem #17

	sDate Date1 = ReadFullDate();
	cout << "\n\n";
	sDate Date2 = ReadFullDate();

	cout << "\nDifference is: "
		<< GetDifferenceInDays(Date1, Date2) << " Day(s).";

	cout << "\nDifference (Including End Day) is: "
		<< GetDifferenceInDays(Date1, Date2, true) << " Day(s).";


	system("pause>0");

	return 0;
}
```

#Problem 18 : Your Age In Days

```
Write a program to read
Calculate you Age in Days

:Please Enter Your Date of Birth

Please enter a Day to check ?
13

Please enter a Month to check
? 12

Please enter a year to check ?
1999


.Your Age is : 8648 Day(s)
```

```cpp
#pragma warning(disable : 4996)

#include <iostream>
#include <string>
#include <iomanip>
#include <ctime>
using namespace std;

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

//Problem #6

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

	int NumberOfDays[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

     return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month - 1];

}

// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

```cpp
// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #17

int GetDifferenceInDays(sDate Date1 , sDate Date2 , bool IncludeEndDay= false)
{
	int Days = 0;

	while (IsDate1BeforeDate2(Date1, Date2))
	{
		Days++;
		Date1 = IncreaseDateByOneDay(Date1);
	}
	return IncludeEndDay ? ++Days : Days;
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}
```

```cpp
// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}



// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

// Problem #18

sDate GetSystemDate()
{
	sDate Date;

	time_t t = time(0);
	tm* now = localtime(&t);

	Date.Year = now->tm_year + 1900;
	Date.Month = now->tm_mon + 1;
	Date.Day = now->tm_mday;

	return Date;
}

int main()
{
	// Problem #18

	cout << "\nPlease Enter Your Date of Birth:\n";

	sDate Date1 = ReadFullDate();

	sDate Date2 = GetSystemDate();

	cout << "\nYour Age is : "
		<< GetDifferenceInDays(Date1, Date2 , true) << " Day(s).";



	system("pause>0");

	return 0;
}
```

#Problem 19 : Diff In Days (Negative Days)

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) : NumberOfDays[Month - 1];

}
// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

Write a program to read Date1 , Date2 and make function to Calculate the Difference in Days

NOTS : if Date2 is less than Date1 Print the Results in Minutes

```
Please enter a Day to check ?
16

Please enter a Month to check
? 8

Please enter a year to check ?
2023


Please enter a Day to check ?
13

Please enter a Month to check
? 12

Please enter a year to check ?
1999


.Difference is: -8647 Day(s)

Difference (Including End Day)
.is: -8648 Day(s)
```

```
// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #19

void SwapDates (sDate &Date1, sDate& Date2)
{
	sDate TempDate;

	TempDate.Year = Date1.Year;
	TempDate.Month = Date1.Month;
	TempDate.Day = Date1.Day;

	Date1.Year = Date2.Year;
	Date1.Month = Date2.Month;
	Date1.Day = Date2.Day;

	Date2.Year = TempDate.Year;
	Date2.Month = TempDate.Month;
	Date2.Day = TempDate.Day;

}

int GetDifferenceInDays(sDate Date1 , sDate Date2 , bool IncludeEndDay = false)
{
	int Days = 0;
	short SwapFlagValue = 1;

	if (! IsDate1BeforeDate2(Date1, Date2))
	{
		SwapDates(Date1, Date2);
		SwapFlagValue = -1;
	}

	while (IsDate1BeforeDate2(Date1, Date2))
	{
		Days++;
		Date1 = IncreaseDateByOneDay(Date1);
	}
	return IncludeEndDay ? ++Days * SwapFlagValue : Days * SwapFlagValue;
}
```

```cpp
// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #12
sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

int main()
{
	// Problem #19

	sDate Date1 = ReadFullDate();
	cout << "\n\n";
	sDate Date2 = ReadFullDate();

	cout << "\nDifference is: "
		<< GetDifferenceInDays(Date1, Date2) << " Day(s).";

	cout << "\nDifference (Including End Day) is: "
		<< GetDifferenceInDays(Date1, Date2, true) << " Day(s).";

	system("pause>0");

	return 0;
}
```

#Problems 20 to 32 : Increase Date Problems

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) : NumberOfDays[Month - 1];

}
// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day ==
NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

Write a program to read a Date and make a functions to Increase Date as follows :

```
Please enter a Day to check ?
18

Please enter a Month to check
? 8

Please enter a year to check ?
2023


:Date After

Adding one day is: -01
19/8/2023

Adding 10 days is: -02
29/8/2023

Adding one week is: -03
5/9/2023

Adding 10 weeks is: -04
14/11/2023

Adding one month is: -05
14/12/2023

Adding 5 months is: -06
14/5/2024

Adding one year is: -07
14/5/2025

Adding 10 Years is: -08
14/5/2035
```

```cpp
// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;

	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

```
Adding 10 Years (faster) is: -09
14/5/2045

Adding one Decade is: -10
14/5/2055

Adding 10 Decades is: -11
14/5/2155

Adding 10 Decade (faster) -12
is: 14/5/2255

Adding One Century is: -13
14/5/2355

Adding One Millennium is: -14
14/5/3355
```

```
// Problem #12
sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

// Problems From 20 To 32

sDate IncreaseDateByXDays(short Days, sDate Date)
{
	for (short i = 1; i <= Days; i++)
	{
		Date = IncreaseDateByOneDay(Date);
	}
	return Date;
}

sDate IncreaseDateByOneWeek(sDate Date)
{
	for (int i = 1; i <= 7; i++)
	{
		Date = IncreaseDateByOneDay(Date);
	}
	return Date;
}
sDate IncreaseDateByXWeeks(short Weeks, sDate Date)
{
	for (short i = 1; i <= Weeks; i++)
	{
		Date = IncreaseDateByOneWeek(Date);
	}
	return Date;
}

sDate IncreaseDateByOneMonth(sDate Date)
{
	if (Date.Month == 12)
	{
		Date.Month = 1;
		Date.Year++;
	}
	else
	{
		Date.Month++;
	}
	//last check day in date should not exceed max days in the current month
		// example if date is 31/1/2022 increasing one month should not be
		31 / 2 / 2022, it should
		// be 28/2/2022
		short NumberOfDaysInCurrentMonth =
		NumberOfDaysInAMonth(Date.Month, Date.Year);
	if (Date.Day > NumberOfDaysInCurrentMonth)
	{
		Date.Day = NumberOfDaysInCurrentMonth;
	}
	return Date;
}
```

```
sDate IncreaseDateByXMonths(short Months, sDate Date)
{
	for (short i = 1; i <= Months; i++)
	{
		Date = IncreaseDateByOneMonth(Date);
	}
	return Date;
}


sDate IncreaseDateByOneYear(sDate Date)
{
	Date.Year++;
	return Date;
}
sDate IncreaseDateByXYears(short Years, sDate Date)
{
	for (short i = 1; i <= Years; i++)
	{
		Date = IncreaseDateByOneYear(Date);
	}
	return Date;
}
sDate IncreaseDateByXYearsFaster(short Years, sDate Date)
{
	Date.Year += Years;
	return Date;
}

sDate IncreaseDateByOneDecade(sDate Date)
{
	//Period of 10 years
	Date.Year += 10;
	return Date;
}

sDate IncreaseDateByXDecades(short Decade, sDate Date)
{
	for (short i = 1; i <= Decade * 10; i++)
	{
		Date = IncreaseDateByOneYear(Date);
	}
	return Date;
}
sDate IncreaseDateByXDecadesFaster(short Decade, sDate Date)
{
	Date.Year += Decade * 10;
	return Date;
}

sDate IncreaseDateByOneCentury(sDate Date)
{
	//Period of 100 years
	Date.Year += 100;
	return Date;
}
sDate IncreaseDateByOneMillennium(sDate Date)
{
	//Period of 1000 years
	Date.Year += 1000;
	return Date;
}
```

```cpp
int main()
{
	// Problems From 20 To 32

	sDate Date1 = ReadFullDate();

	 cout << "\nDate After: \n";

	 Date1 = IncreaseDateByOneDay(Date1);
	 cout << "\n01-Adding one day is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;
	 Date1 = IncreaseDateByXDays(10, Date1);
	 cout << "\n02-Adding 10 days is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

	 Date1 = IncreaseDateByOneWeek(Date1);
	 cout << "\n03-Adding one week is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;
	 Date1 = IncreaseDateByXWeeks(10, Date1);
	 cout << "\n04-Adding 10 weeks is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

	 Date1 = IncreaseDateByOneMonth(Date1);
	 cout << "\n05-Adding one month is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;
	 Date1 = IncreaseDateByXMonths(5, Date1);
	 cout << "\n06-Adding 5 months is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

	 Date1 = IncreaseDateByOneYear(Date1);
	 cout << "\n07-Adding one year is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;
	 Date1 = IncreaseDateByXYears(10, Date1);
	 cout << "\n08-Adding 10 Years is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;
	 Date1 = IncreaseDateByXYearsFaster(10, Date1);
	 cout << "\n09-Adding 10 Years (faster) is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

	 Date1 = IncreaseDateByOneDecade(Date1);
	 cout << "\n10-Adding one Decade is: "
	 << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;
	 Date1 = IncreaseDateByXDecades(10, Date1);
	 cout << "\n11-Adding 10 Decades is: "
	 	<< Date1.Day << "/" << Date1.Month << "/" << Date1.Year;
	 Date1 = IncreaseDateByXDecadesFaster(10, Date1);
	 cout << "\n12-Adding 10 Decade (faster) is: "
	 	<< Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

	 Date1 = IncreaseDateByOneCentury(Date1);
	 cout << "\n13-Adding One Century is: "
	 	<< Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

	 Date1 = IncreaseDateByOneMillennium(Date1);
	 cout << "\n14-Adding One Millennium is: "
	 	<< Date1.Day << "/" << Date1.Month << "/" << Date1.Year;


	system("pause>0");

	return 0;
}
```

#Problems 33 to 46 : Decrease Date Problems

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3
bool IsLeapYear(short Year)
{
	return (Year % 4 == 0 && Year % 100 != 0) ||
(Year % 400 == 0);
}

//Problem #6
short NumberOfDaysInAMonth(short Month, short Year)
{
	if (Month < 1 || Month > 12)
		return  0;

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) : NumberOfDays[Month - 1];

}
// Problem #15
bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day ==
NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

// Problems From 33 To 46
sDate DecreaseDateByOneDay(sDate Date)
{

	if (Date.Day == 1)
	{
		if (Date.Month == 1)
		{
			Date.Month = 12;
			Date.Day = 31;
			Date.Year--;
		}
		else
		{
			Date.Month--;
			Date.Day = NumberOfDaysInAMonth(Date.Month, Date.Year);
```

Write a program to read a Date and make a functions to Decrease Date as follows :

```
Please enter a Day to check ?
19

Please enter a Month to check
? 8

Please enter a year to check ?
2023

:Date After


Subtracting one day is: -01
18/8/2023

Subtracting 10 days is: -02
8/8/2023

Subtracting one week is: -03
1/8/2023

Subtracting 10 weeks is: -04
23/5/2023

Subtracting one month is: -05
23/4/2023

Subtracting 5 months is: -06
23/11/2022

Subtracting one year is: -07
23/11/2021

Subtracting 10 Years is: -08
23/11/2011
```

```
                }
        }
        else
        {
                Date.Day--;
        }
        return Date;
}

sDate DecreaseDateByXDays(short Days, sDate Date)
{
        for (short i = 1; i <= Days; i++)
        {
                Date = DecreaseDateByOneDay(Date);
        }
        return Date;
}

sDate DecreaseDateByOneWeek(sDate Date)
{
        for (int i = 1; i <= 7; i++)
        {
                Date = DecreaseDateByOneDay(Date);
        }
        return Date;
}

sDate DecreaseDateByXWeeks(short Weeks, sDate Date)
{
        for (short i = 1; i <= Weeks; i++)
        {
                Date = DecreaseDateByOneWeek(Date);
        }
        return Date;
}

sDate DecreaseDateByOneMonth(sDate Date)
{
        if (Date.Month == 1)
        {
                Date.Month = 12;
                Date.Year--;
        }
        else
        {
                Date.Month--;
        }
                short NumberOfDaysInCurrentMonth =
                NumberOfDaysInAMonth(Date.Month, Date.Year);
        if (Date.Day > NumberOfDaysInCurrentMonth)
        {
                Date.Day = NumberOfDaysInCurrentMonth;
        }
        return Date;
}
sDate DecreaseDateByXMonths(short Months, sDate Date)
{
        for (short i = 1; i <= Months; i++)
        {
                Date = DecreaseDateByOneMonth(Date);
        }
        return Date;
}
```

```
Subtracting 10 Years -09
(faster) is: 23/11/2001

Subtracting one Decade is: -10
23/11/1991

Subtracting 10 Decades is: -11
23/11/1891

Subtracting 10 Decade -12
(faster) is: 23/11/1791

Subtracting One Century is: -13
23/11/1691

Subtracting One -14
Millennium is: 23/11/691
```

```
sDate DecreaseDateByOneYear(sDate Date)
{
	Date.Year--;
	return Date;
}
sDate DecreaseDateByXYears(short Years, sDate Date)
{
	for (short i = 1; i <= Years; i++)
	{
		Date = DecreaseDateByOneYear(Date);
	}
	return Date;
}
sDate DecreaseDateByXYearsFaster(short Years, sDate Date)
{
	Date.Year -= Years;
	return Date;
}

sDate DecreaseDateByOneDecade(sDate Date)
{
	//Period of 10 years
	Date.Year -= 10;
	return Date;
}
sDate DecreaseDateByXDecades(short Decade, sDate Date)
{
	for (short i = 1; i <= Decade * 10; i++)
	{
		Date = DecreaseDateByOneYear(Date);
	}
	return Date;
}
sDate DecreaseDateByXDecadesFaster(short Decade, sDate Date)
{
	Date.Year -= Decade * 10;
	return Date;
}

sDate DecreaseDateByOneCentury(sDate Date)
{
	//Period of 100 years
	Date.Year -= 100;
	return Date;
}
sDate DecreaseDateByOneMillennium(sDate Date)
{
	//Period of 1000 years
	Date.Year -= 1000;
	return Date;
}


int main()
{
	// Problems From 33 To 46

	sDate Date1 = ReadFullDate();

	 cout << "\nDate After: \n";
```

```
        Date1 = DecreaseDateByOneDay(Date1);
        cout << "\n01-Subtracting one day is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByXDays(10, Date1);
        cout << "\n02-Subtracting 10 days is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByOneWeek(Date1);
        cout << "\n03-Subtracting one week is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByXWeeks(10, Date1);
        cout << "\n04-Subtracting 10 weeks is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByOneMonth(Date1);
        cout << "\n05-Subtracting one month is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByXMonths(5, Date1);
        cout << "\n06-Subtracting 5 months is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByOneYear(Date1);
        cout << "\n07-Subtracting one year is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByXYears(10, Date1);
        cout << "\n08-Subtracting 10 Years is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByXYearsFaster(10, Date1);
        cout << "\n09-Subtracting 10 Years (faster) is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByOneDecade(Date1);
        cout << "\n10-Subtracting one Decade is: "
        << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByXDecades(10, Date1);
        cout << "\n11-Subtracting 10 Decades is: "
              << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByXDecadesFaster(10, Date1);
        cout << "\n12-Subtracting 10 Decade (faster) is: "
              << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByOneCentury(Date1);
        cout << "\n13-Subtracting One Century is: "
              << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;

        Date1 = DecreaseDateByOneMillennium(Date1);
        cout << "\n14-Subtracting One Millennium is: "
              << Date1.Day << "/" << Date1.Month << "/" << Date1.Year;


       system("pause>0");

       return 0;
}
```

#Problems 47 to 53 : More Date Problems

Write a program to read a date and make functions as follows :

Today is Sun , 20/8/2023

? Is it End of Week

. No , it is Not End of Week

? Is it Weekend

No , today is : Sun Not a Week end

? Is it Business Day

. Yes , it is a Business Day

Days Until end of Week : 6 .Days

Days Until end of Month : 12 .Days

Days Until end of Month : 134 .Days

```
#pragma warning(disable : 4996)

#include <iostream>
#include <string>
#include <iomanip>
#include <ctime>
using namespace std;

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

     return (Month == 2) ? (IsLeapYear(Year) ? 29 :
28) : NumberOfDays[Month - 1];

}

// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}
```

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

// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #17

int GetDifferenceInDays(sDate Date1 , sDate Date2 , bool IncludeEndDay= false)
{
	int Days = 0;

	while (IsDate1BeforeDate2(Date1, Date2))
	{
		Days++;
		Date1 = IncreaseDateByOneDay(Date1);
	}
	return IncludeEndDay ? ++Days : Days;
}

// Problem #7

short DayOfWeekOrder(short Day, short Month, short Year)
{
	short a, y, m;
	a = (14 - Month) / 12;
	y = Year - a;
	m = Month + (12 * a) - 2;
	// Gregorian:
	//0:sun, 1:Mon, 2:Tue...etc.
	return (Day + y + (y / 4) - (y / 100) + (y / 400) + ((31 * m) / 12)) % 7;
}

short DayOfWeekOrder(sDate Date)
{
	return DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
}
```

```
string DayShortName(short DayOfWeekOrder)
{
	string arrDayNames[7] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder];
}
// Problems From 47 To 53

bool IsEndOfWeek(sDate Date)
{
	return DayOfWeekOrder(Date) == 6;
}

bool IsWeekend(sDate Date)
{
	// Weekends are Fri and Sat
	short DayIndex = DayOfWeekOrder(Date);
	return ( DayIndex == 6 || DayIndex == 5 );
}

bool IsBusinessDay(sDate Date)
{
	// Weekends are Sun , Mon , Tue , Wed and Thur

	/*
	short DayIndex = DayOfWeekOrder(Date);
	return ( DayIndex >= 0 && DayIndex <= 4 );
	*/

	return ! IsWeekend(Date);
}

short DaysUntilTheEndOfWeek(sDate Date)
{
	return 6 - DayOfWeekOrder(Date) ;
}

short DaysUntilTheEndOfMonth(sDate Date)
{
	sDate EndOfMonthDate;

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

	return GetDifferenceInDays(Date , EndOfMonthDate , true );
}

short DaysUntilTheEndOfYear(sDate Date)
{
	sDate EndOfMonthDate;

	EndOfMonthDate.Day = 31;
	EndOfMonthDate.Month = 12;
	EndOfMonthDate.Year = Date.Year;

	return GetDifferenceInDays(Date, EndOfMonthDate, true);
}
```

```
// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

// Problem #18

sDate GetSystemDate()
{
	sDate Date;

	time_t t = time(0);
	tm* now = localtime(&t);

	Date.Year = now->tm_year + 1900;
	Date.Month = now->tm_mon + 1;
	Date.Day = now->tm_mday;

	return Date;
}
```

```cpp
int main()
{
	// Problems From 47 To 53

	sDate Date;


	Date = GetSystemDate();


	cout << "\nToday is " << DayShortName(DayOfWeekOrder(Date)) << " , "
		<< Date.Day << "/" << Date.Month << "/" << Date.Year << endl;

	//--------------------
	cout << "\nIs it End of Week ? \n";

	if (IsEndOfWeek(Date))
		cout << 	"Yes , it is Saturday , it's of Week .";
	else
		cout << 	"No , it is Not End of Week .";
	//--------------------

	cout << "\n\nIs it Weekend ? \n";

	if (IsWeekend(Date))
		cout << 	"Yes , it is a Week end .";
	else
		cout << 	"No , today is : "<< DayShortName(DayOfWeekOrder(Date))
<< " Not a Week end";
	//--------------------

	cout << "\n\nIs it Business Day ? \n";

	if (IsBusinessDay(Date))
		cout << 	"Yes , it is a Business Day .";
	else
		cout << 	"No , it is Not Business Day .";
	//--------------------

	cout << "\n\nDays Until end of Week : "
		<< DaysUntilTheEndOfWeek(Date) << " Days.";
	//--------------------

	cout << "\nDays Until end of Month : "
		<< DaysUntilTheEndOfMonth(Date) << " Days.";
	//--------------------

	cout << "\nDays Until end of Month : "
		<< DaysUntilTheEndOfYear(Date) << " Days.";
	//--------------------



	system("pause>0");

	return 0;
}
```

#Problem 54 : Calculate Vacation Days

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) : NumberOfDays[Month - 1];

}
// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

Write a program to read Vacation Period DateFrom and DateTo and Make function to Calculate the actual Vacation Days

```
: Vacations Starts

Please enter a Day to check ? 1

Please enter a Month to check
? 8

Please enter a year to check ?
2023


: Vacations Ends

Please enter a Day to check ?
21

Please enter a Month to check
? 8

Please enter a year to check ?
2023


Vacation From : Tue , 1/8/2023

Vacation End : Mon , 21/8/2023

Actual Vacations Days is : 14
```

```
// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #7

short DayOfWeekOrder(short Day, short Month, short Year)
{
	short a, y, m;
	a = (14 - Month) / 12;
	y = Year - a;
	m = Month + (12 * a) - 2;
	// Gregorian:
	//0:sun, 1:Mon, 2:Tue...etc.
	return (Day + y + (y / 4) - (y / 100) + (y / 400) + ((31 * m) / 12)) % 7;
}

short DayOfWeekOrder(sDate Date)
{
	return DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
}


string DayShortName(short DayOfWeekOrder)
{
	string arrDayNames[7] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder];
}

// Problems From 33 To 46

bool IsWeekend(sDate Date)
{
	// Weekends are Fri and Sat
	short DayIndex = DayOfWeekOrder(Date);
	return ( DayIndex == 6 || DayIndex == 5 );
}
```

```
bool IsBusinessDay(sDate Date)
{
	// Weekends are Sun , Mon , Tue , Wed and Thru

	/*
	short DayIndex = DayOfWeekOrder(Date);
	return ( DayIndex >= 0 && DayIndex <= 4 );
	*/

	return ! IsWeekend(Date);
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}
```

```
// Problem #54

short CalculateVacationDays(sDate DateFrom, sDate DateTo)
{
	short DaysCount = 0;

	while (IsDate1BeforeDate2(DateFrom, DateTo))
	{
		if (IsBusinessDay(DateFrom))
			DaysCount++;
		DateFrom = IncreaseDateByOneDay(DateFrom);

	}

	return DaysCount;
}

int main()
{
	// Problem #54

	cout << "\nVacations Starts : \n";
	sDate DateFrom = ReadFullDate();

	cout << "\nVacations Ends : \n";
	sDate DateTo = ReadFullDate();

	cout << "\n\nVacation From : " << DayShortName(DayOfWeekOrder(DateFrom))
<< " , "
		<< DateFrom.Day << "/" << DateFrom.Month << "/" << DateFrom.Year <<
endl;

	cout << "\n\nVacation End : " << DayShortName(DayOfWeekOrder(DateTo)) << "
, "
		<< DateTo.Day << "/" << DateTo.Month << "/" << DateTo.Year << endl;

	cout << "\n\nActual Vacations Days is : " <<
CalculateVacationDays(DateFrom, DateTo);

	system("pause>0");

	return 0;
}
```

#Problem 55 : Calculate Vacation Return Date

Write a program to read Vacation Start DateFrom and VacationDays , then make a function to Calculate the Vacation Return Date

```
:Vacation Starts

Please enter a Day to check ? 1

Please enter a Month to check
? 1

Please enter a year to check ?
2023


Please enter vacation days? 23


Return Date: Wed , 1/2/2023
```

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

//Problem #6

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

	int NumberOfDays[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month -
1];

}
// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

```
// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}

// Problem #7

short DayOfWeekOrder(short Day, short Month, short Year)
{
	short a, y, m;
	a = (14 - Month) / 12;
	y = Year - a;
	m = Month + (12 * a) - 2;
	// Gregorian:
	//0:sun, 1:Mon, 2:Tue...etc.
	return (Day + y + (y / 4) - (y / 100) + (y / 400) + ((31 * m) / 12)) % 7;
}

short DayOfWeekOrder(sDate Date)
{
	return DayOfWeekOrder(Date.Day, Date.Month, Date.Year);
}


string DayShortName(short DayOfWeekOrder)
{
	string arrDayNames[7] = { "Sun","Mon","Tue","Wed","Thu","Fri","Sat" };

	return arrDayNames[DayOfWeekOrder];
}

bool IsWeekEnd(sDate Date)
{
	// Weekends are Fri and Sat
	short DayIndex = DayOfWeekOrder(Date);
	return ( DayIndex == 6 || DayIndex == 5 );
}
```

```
bool IsBusinessDay(sDate Date)
{
	// Weekends are Sun , Mon , Tue , Wed and Thur

	/*
	short DayIndex = DayOfWeekOrder(Date);
	return ( DayIndex >= 0 && DayIndex <= 4 );
	*/

	return ! IsWeekEnd(Date);
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}

// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}
```

```
// Problem #55

sDate CalculateVacationReturnDate(sDate DateFrom, short VacationDays)
{

	short WeekEndCounter = 0;

	//in case the data  is weekend keep adding one day util you reach business
day
	//we get rid of all weekends before the first business day
	while (IsWeekEnd(DateFrom))
	{
		DateFrom = IncreaseDateByOneDay(DateFrom);
	}

	//here we increase the vacation dates to add all weekends to it.

	for (short i = 1; i <= VacationDays + WeekEndCounter; i++)
	{

		if (IsWeekEnd(DateFrom))
			WeekEndCounter++;

		DateFrom = IncreaseDateByOneDay(DateFrom);
	}

	//in case the return date is week end keep adding one day util you reach
business day
	while (IsWeekEnd(DateFrom))
	{
		DateFrom = IncreaseDateByOneDay(DateFrom);
	}

	return DateFrom;
}

short ReadVacationDays()
{
	short Days;
	cout << "\nPlease enter vacation days? ";
	cin >> Days;
	return Days;
}

int main()
{
	// Problem #55

   cout << "\nVacation Starts: ";
    sDate DateFrom = ReadFullDate();

    short VacationDays = ReadVacationDays();

    sDate ReturnDate = CalculateVacationReturnDate(DateFrom, VacationDays);

    cout << "\n\nReturn Date: " << DayShortName(DayOfWeekOrder(ReturnDate)) << "
, "
	<< ReturnDate.Day << "/" << ReturnDate.Month << "/" << ReturnDate.Year <<
endl;
	system("pause>0");

	return 0;
}
```

#Problem 56 : Is Date1 After Date2

Write a program to read Date1 & Date2 and check if Date1 is after Date2 or Not

```
: Enter Date1

Please enter a Day to check ? 1

Please enter a Month to check
? 1

Please enter a year to check ?
2023


: Enter Date2

Please enter a Day to check ? 1

Please enter a Month to check
? 1

Please enter a year to check ?
2000


.Yes , Date1 is After Date2
```

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

// Problem #11

struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #13

bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

// Problem #14

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

// Problem #56

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

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}
```

```cpp
// Problem #5

short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2

short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}


int main()
{
	// Problem #56

	cout << "\nEnter Date1 : ";
	sDate Date1 = ReadFullDate();


	cout << "\nEnter Date2 : ";
	sDate Date2 = ReadFullDate();

	if (IsDate1AfterDate2(Date1, Date2))
		cout << "\nYes , Date1 is After Date2.";
	else
		cout << "\nNo , Date1 is NOT After Date2.";


	system("pause>0");

	return 0;
}
```

#Problem 57 : Compare Date Function

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #13
bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

// Problem #14
bool IsDate1EqualDate2(sDate Date1, sDate Date2)
{
	return (Date1.Year == Date2.Year ) ? (( 
Date1.Month == Date2.Month ) ? 
		((Date1.Day == Date2.Day) ? true : 
false ) : false ) : false ;
}

// Problem #56
bool IsDate1AfterDate2(sDate Date1, sDate Date2)
{
	return (! IsDate1BeforeDate2(Date1, Date2) 
&& ! IsDate1EqualDate2(Date1, Date2));
}

// Problem #57
enum enDateCompare {Before = -1 , Equal = 0 , After 
= 1};

enDateCompare CompareDates(sDate Date1 , sDate 
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;
}
```

Write a program to read Date1 & Date2 and write a function to compare Dates , it should return :

*** -1 Before**

*** 0 Equal**

*** 1 After**

```
:Enter Date1

Please enter a Day to check ? 1

Please enter a Month to check
? 1

Please enter a year to check ?
2000


:Enter Date2

Please enter a Day to check ? 1

Please enter a Month to check
? 1

Please enter a year to check ?
2000


Compare Result = 0
```

```cpp
// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}

// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}
int main()
{
	// Problem #57

	cout << "\nEnter Date1:";
	sDate Date1 = ReadFullDate();

	cout << "\nEnter Date2:";
	sDate Date2 = ReadFullDate();

	cout << "\nCompare Result = " << CompareDates(Date1, Date2);

		cout << "\nNo , Date1 is NOT After Date2.";


	system("pause>0");

	return 0;
}
```

#Problem 58 : Is Overlap Periods

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #58

struct stPeriod
{
	stDate StartDate;
	stDate EndDate;
};

// Problem #13
bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

// Problem #14
bool IsDate1EqualDate2(sDate Date1, sDate Date2)
{
	return (Date1.Year == Date2.Year ) ? ((
Date1.Month == Date2.Month ) ? 
		((Date1.Day == Date2.Day) ? true : 
false ) : false ) : false ;
}

// Problem #56
bool IsDate1AfterDate2(sDate Date1, sDate Date2)
{
	return (! IsDate1BeforeDate2(Date1, Date2) 
&& ! IsDate1EqualDate2(Date1, Date2));
}

// Problem #57
enum enDateCompare {Before = -1 , Equal = 0 , After = 1};
```

Write a program to read Two Periods and check if they Overlap OR NOT ?

```
:Enter Period 1

:Enter Start Date


Please enter a Day to check ? 1

Please enter a Month to check 
? 1

Please enter a year to check ? 
2023


:Enter End Date

Please enter a Day to check ? 
10

Please enter a Month to check 
? 1

Please enter a year to check ? 
2023
```

```
enDateCompare CompareDates(sDate Date1 , sDate
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;
}

// Problem #58

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;
}

// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}
```

```
:Enter Period 2

:Enter Start Date


Please enter a Day to check ? 5

Please enter a Month to check
? 1

Please enter a year to check ?
2023


:Enter End Date


Please enter a Day to check ?
15

Please enter a Month to check
? 1

Please enter a year to check ?
2023


Yes, Periods Overlap
```

```cpp
// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

// Problem #58


stPeriod ReadPeriod()
{
	stPeriod Period;

	cout << "\nEnter Start Date:\n";
	Period.StartDate = ReadFullDate();

	cout << "\nEnter End Date:\n";
	Period.EndDate = ReadFullDate();

	return Period;
}

int main()
{
	// Problem #58

	cout << "\nEnter Period 1:";
	stPeriod Period1 = ReadPeriod();

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

	if (IsOverlapPeriods(Period1, Period2))
		cout << "\nYes, Periods Overlap\n";
	else
		cout << "\nNo, Periods do not Overlap\n";


	system("pause>0");

	return 0;
}
```

#Problem 59 : Period Length In Days

Write a program to read a Period and Calculate Period Length In Days ?

```
:Enter Period 1

:Enter Start Date

Please enter a Day to check ? 8

Please enter a Month to check
? 8

Please enter a year to check ?
2023


:Enter End Date

Please enter a Day to check ?
22

Please enter a Month to check
? 8

Please enter a year to check ?
2023


Period Length is: 14

Period Length (Including End
Date) is: 15
```

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #58

struct stPeriod
{
	stDate StartDate;
	stDate EndDate;
};

// Problem #13
bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

// Problem #14
bool IsDate1EqualDate2(sDate Date1, sDate Date2)
{
	return (Date1.Year == Date2.Year ) ? ((
Date1.Month == Date2.Month ) ?
		((Date1.Day == Date2.Day) ? true :
false ) : false ) : false ;
}

// Problem #56
bool IsDate1AfterDate2(sDate Date1, sDate Date2)
{
	return (! IsDate1BeforeDate2(Date1, Date2)
&& ! IsDate1EqualDate2(Date1, Date2));
}

// Problem #3

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

```
//Problem #6

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

	int NumberOfDays[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month -
1];

}
// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day == NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}
```

```
// Problem #19

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

	int Days = 0;
	short SwapFlagValue = 1;

	if (! IsDate1BeforeDate2(Date1, Date2))
	{
		SwapDates(Date1, Date2);
		SwapFlagValue = -1;
	}

	while (IsDate1BeforeDate2(Date1, Date2))
	{
		Days++;
		Date1 = IncreaseDateByOneDay(Date1);
	}
	return IncludeEndDay ? ++Days * SwapFlagValue : Days * SwapFlagValue;
}

// Problem #59

int PeriodLengthInDays(stPeriod Period, bool IncludeEndDate = false)
{
	return GetDifferenceInDays(Period.StartDate, Period.EndDate,
IncludeEndDate);
}
// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
```

```cpp
// Problem #12

sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

// Problem #58


stPeriod ReadPeriod()
{
	stPeriod Period;

	cout << "\nEnter Start Date:\n";
	Period.StartDate = ReadFullDate();

	cout << "\nEnter End Date:\n";
	Period.EndDate = ReadFullDate();

	return Period;
}

int main()
{
	// Problem #59

	cout << "\nEnter Period 1:";
	stPeriod Period1 = ReadPeriod();

	cout << "\nPeriod Length is: " << PeriodLengthInDays(Period1);
	cout << "\nPeriod Length (Including End Date) is: "
		<< PeriodLengthInDays(Period1, true);



	system("pause>0");

	return 0;
}
```

#Problem 60 : Is Date Within Period

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #58

struct stPeriod
{
	stDate StartDate;
	stDate EndDate;
};

// Problem #13
bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

// Problem #14
bool IsDate1EqualDate2(sDate Date1, sDate Date2)
{
	return (Date1.Year == Date2.Year ) ? ((
Date1.Month == Date2.Month ) ?
		((Date1.Day == Date2.Day) ? true :
false ) : false ) : false ;
}

// Problem #56
bool IsDate1AfterDate2(sDate Date1, sDate Date2)
{
	return (! IsDate1BeforeDate2(Date1, Date2)
&& ! IsDate1EqualDate2(Date1, Date2));
}

// Problem #57
enum enDateCompare {Before = -1 , Equal = 0 , After
= 1};
```

Write a program to read Period and Date , then check if Date is within this Period OR NOT ?

```
: Enter Period

:Enter Start Date

Please enter a Day to check ? 1

Please enter a Month to check
? 8

Please enter a year to check ?
2023


:Enter End Date

Please enter a Day to check ?
31

Please enter a Month to check
? 8

Please enter a year to check ?
2023


:Enter Date to check

Please enter a Day to check ?
23

Please enter a Month to check
? 8

Please enter a year to check ?
2023


Yes, Date is within period
```

```cpp
enDateCompare CompareDates(sDate Date1 , sDate 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;
}

// Problem #60

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

// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}

// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}


// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}


// Problem #12

sDate ReadFullDate()
```

```
{
	sDate Date;

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

	return Date;
}

// Problem #58


stPeriod ReadPeriod()
{
	stPeriod Period;

	cout << "\nEnter Start Date:\n";
	Period.StartDate = ReadFullDate();

	cout << "\nEnter End Date:\n";
	Period.EndDate = ReadFullDate();

	return Period;
}

int main()
{
	// Problem #60

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

	cout << "\nEnter Date to check:\n";
	stDate Date = ReadFullDate();

	if (isDateInPeriod(Date, Period))
		cout << "\nYes, Date is within period\n";
	else
		cout << "\nNo, Date is NOT within period\n";


	system("pause>0");

	return 0;
}
```

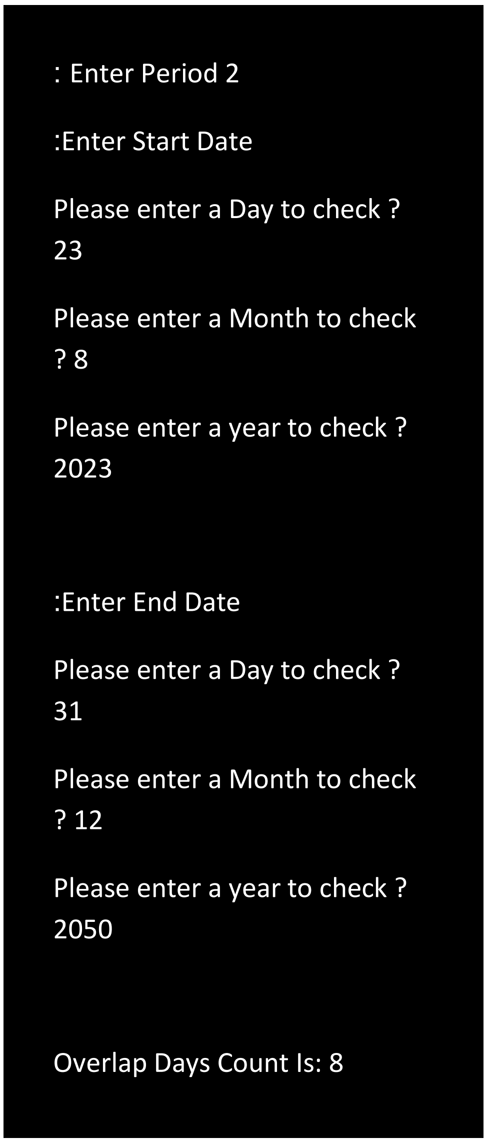
#Problem 61: Count Overlap Days

Write a program to read Two Periods then Count Overlap Days ?

```
: Enter Period 1

:Enter Start Date

Please enter a Day to check ? 1

Please enter a Month to check
? 8

Please enter a year to check ?
2023


:Enter End Date

Please enter a Day to check ?
31

Please enter a Month to check
? 8

Please enter a year to check ?
2023


: Enter Period 2

:Enter Start Date
```

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #58

struct stPeriod
{
	stDate StartDate;
	stDate EndDate;
};

// Problem #13
bool IsDate1BeforeDate2(sDate Date1, sDate 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);
}

// Problem #14
bool IsDate1EqualDate2(sDate Date1, sDate Date2)
{
	return (Date1.Year == Date2.Year ) ? ((
Date1.Month == Date2.Month ) ?
		((Date1.Day == Date2.Day) ? true :
false ) : false ) : false ;
}

// Problem #56
bool IsDate1AfterDate2(sDate Date1, sDate Date2)
{
	return (! IsDate1BeforeDate2(Date1, Date2)
&& ! IsDate1EqualDate2(Date1, Date2));
}

// Problem #3

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

```
//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29
: 28) : NumberOfDays[Month - 1];

}
// Problem #15

bool IsLastDayInMonth(sDate Date)
{
	return (Date.Day ==
NumberOfDaysInAMonth(Date.Month, Date.Year));
}

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

// Problem #16

sDate IncreaseDateByOneDay(sDate 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;
}
```



```
: Enter Period 2

:Enter Start Date

Please enter a Day to check ?
23

Please enter a Month to check
? 8

Please enter a year to check ?
2023


:Enter End Date

Please enter a Day to check ?
31

Please enter a Month to check
? 12

Please enter a year to check ?
2050


Overlap Days Count Is: 8
```

```
// Problem #19

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

	int Days = 0;
	short SwapFlagValue = 1;

	if (! IsDate1BeforeDate2(Date1, Date2))
	{
		SwapDates(Date1, Date2);
		SwapFlagValue = -1;
	}

	while (IsDate1BeforeDate2(Date1, Date2))
	{
		Days++;
		Date1 = IncreaseDateByOneDay(Date1);
	}
	return IncludeEndDay ? ++Days * SwapFlagValue : Days * SwapFlagValue;
}

// Problem #57

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;
}


// Problem #58


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;
}
```

```
// Problem #59

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

// Problem #60

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

// Problem #61

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;
}

// Problem #7

short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}
```

```cpp
// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}

// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
// Problem #12
sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

// Problem #58
stPeriod ReadPeriod()
{
	stPeriod Period;

	cout << "\nEnter Start Date:\n";
	Period.StartDate = ReadFullDate();

	cout << "\nEnter End Date:\n";
	Period.EndDate = ReadFullDate();

	return Period;
}

int main()
{
	// Problem #61

	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;
}
```

#Problem 62 : Validate Date

Write a program to read Datr and write a function to Validate this Date

```
Please enter a Day to check ?
33

Please enter a Month to check
? 15

Please enter a year to check ?
2023


No, Date is a NOT validate  date
```

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};

// Problem #3

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

//Problem #6

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

	int NumberOfDays[12] = {
31,28,31,30,31,30,31,31,30,31,30,31 };

	return (Month == 2) ? (IsLeapYear(Year) ? 29 : 28) : NumberOfDays[Month -
1];

}
// Problem #62

bool IsValidDate(stDate Date)
{
	if (Date.Day < 1 || Date.Day > 31)
		return false;

	if (Date.Month < 1 || Date.Month>12)
		return false;

	if (Date.Month == 2)
	{
		if
			(IsLeapYear(Date.Year))
		{
			if (Date.Day > 29)
				return false;
		}
		else
		{
			if (Date.Day > 28)
				return false;
		}
	}
	short DaysInMonth = NumberOfDaysInAMonth(Date.Month, Date.Year);

	if (Date.Day > DaysInMonth)
		return false;

	return true;

}
// Problem #7
short ReadDay()
{
	short Day = 0;

	cout << "\nPlease enter a Day to check ? ";
	cin >> Day;
	return Day;
}
// Problem #5
short ReadMonth()
{
	short Month = 0;

	cout << "\nPlease enter a Month to check ? ";
	cin >> Month;

	return Month;
}
// Problem #2
short ReadYear()
{
	short Year = 0;

	cout << "\nPlease enter a year to check ? ";
	cin >> Year;

	return Year;
}
// Problem #12
sDate ReadFullDate()
{
	sDate Date;

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

	return Date;
}

int main()
{
	// Problem #62

	stDate Date1 = ReadFullDate();

	if (IsValidDate(Date1))
		cout << "\nYes, Date is a validate date.\n";
	else
		cout << "\nNo, Date is a NOT validate date\n";

	system("pause>0");

	return 0;  }
```

#Problems 63 and 64 : Read / Print : Date String

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

// Problem #11
struct sDate
{
	short Year;
	short Month;
	short Day;
};
// Problem #63 and #64

vector<string> SplitString(string S1, string
Delim)
{
	vector<string> vString;
	short pos = 0;
	string sWord; // define a string variable

	// use find() function to get the position
of the delimiters

	while ((pos = S1.find(Delim)) !=
std::string::npos)
	{
		sWord = S1.substr(0, pos); // store
the word
		if (sWord != "")
		{
			vString.push_back(sWord);
		}
		S1.erase(0, pos + Delim.length());
	}
	if (S1 != "")
	{
		vString.push_back(S1); // it adds
last word of the string.
	}
	return vString;
}

string DateToString(stDate Date)
{
	return  to_string(Date.Day) + "/" +
to_string(Date.Month) + "/" +
to_string(Date.Year);
}
```

Write a program to read
*** Read Date string**
*** Convert it to date structure**
***Print Day , Month , Year Separately**
*** Then Convert Date structure to String and Print it on the Screen**

Note : write the following functions :
*** StringToDate**
***DateToString**

```
Please Enter Date dd/mm/yyyy?
24/8/2023


Day:24

Month:8

Year:2023


You Entered: 24/8/2023
```

```cpp
stDate StringToDate(string DateString)
{
	stDate Date;
	vector <string> vDate;

	vDate = SplitString(DateString, "/");

	Date.Day = stoi(vDate[0]);
	Date.Month = stoi(vDate[1]);
	Date.Year = stoi(vDate[2]);

	return Date;
}

string ReadStringDate(string Message)
{
	string DateString;
	cout << Message;
	getline(cin >> ws, DateString);

	return DateString;
}

int main()
{
	// Problem #63 and #64

	string DateString = ReadStringDate("\nPlease Enter Date dd/mm/yyyy? ");

	stDate Date = StringToDate(DateString);

	cout << "\nDay:" << Date.Day << endl;
	cout << "Month:" << Date.Month << endl;
	cout << "Year:" << Date.Year << endl;

	cout << "\nYou Entered: " << DateToString(Date) << "\n";



	system("pause>0");

	return 0;
}
```

#Problems 65: Format Date

Write a program to read Date and write a Function to format that Date

```
Please Enter Date dd/mm/yyyy?
26/8/2023


26/8/2023

2023/26/8

8/26/2023

8-26-2023

26-8-2023

Day: 26, Month: 8, Year: 2023
```

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

// Problem #11
struct sDate
{
        short Year;
        short Month;
        short Day;
};
// Problem #63 and #64

vector<string> SplitString(string S1, string
Delim)
{
        vector<string> vString;
        short pos = 0;
        string sWord; // define a string variable

        // use find() function to get the position
of the delimiters

        while ((pos = S1.find(Delim)) !=
std::string::npos)
        {
                sWord = S1.substr(0, pos); // store
the word
                if (sWord != "")
                {
                        vString.push_back(sWord);
                }
                S1.erase(0, pos + Delim.length());
        }
        if (S1 != "")
        {
                vString.push_back(S1); // it adds last word of the string.
        }
        return vString;
}

// Problem #65

string ReplaceWordInString(string S1, string StringToReplace, string sRepalceTo)
{
        short pos = S1.find(StringToReplace);

        while (pos != std::string::npos)
        {
                S1 = S1.replace(pos, StringToReplace.length(), sRepalceTo);

                pos = S1.find(StringToReplace);//find next
        }
        return S1;
}
```

```
string DateToString(stDate Date)
{
	return  to_string(Date.Day) + "/" + to_string(Date.Month) + "/" +
to_string(Date.Year);
}

stDate StringToDate(string DateString)
{
	stDate Date;
	vector <string> vDate;

	vDate = SplitString(DateString, "/");

	Date.Day = stoi(vDate[0]);
	Date.Month = stoi(vDate[1]);
	Date.Year = stoi(vDate[2]);

	return Date;
}

// Problem #65

string FormateDate(stDate Date, string DateFormat = "dd/mm/yyyy")
{
	string FormattedDateString = "";

	FormattedDateString = ReplaceWordInString(DateFormat, "dd",
to_string(Date.Day));
	FormattedDateString = ReplaceWordInString(FormattedDateString, "mm",
to_string(Date.Month));
	FormattedDateString = ReplaceWordInString(FormattedDateString, "yyyy",
to_string(Date.Year));

	return  FormattedDateString;
}

string ReadStringDate(string Message)
{
	string DateString;
	cout << Message;
	getline(cin >> ws, DateString);

	return DateString;
}

int main()
{
	// Problem #65

	string DateString = ReadStringDate("\nPlease Enter Date dd/mm/yyyy? ");

	stDate Date = StringToDate(DateString);

	cout << "\n" << FormateDate(Date) << "\n";
	cout << "\n" << FormateDate(Date, "yyyy/dd/mm") << "\n";
	cout << "\n" << FormateDate(Date, "mm/dd/yyyy") << "\n";
	cout << "\n" << FormateDate(Date, "mm-dd-yyyy") << "\n";
	cout << "\n" << FormateDate(Date, "dd-mm-yyyy") << "\n";
	cout << "\n" << FormateDate(Date, "Day:dd, Month:mm, Year:yyyy") << "\n";

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