Lecture Summary

Data types

Recall that a *data type* is set of values and set of operations on those values. It determines the possible values that can be stored in a variable and the operations that can be performed on that variable. It also determines the amount of memory required to store the value and the way the value is stored in the memory. Some commonly used data types in C++ are following[1]:

- **int** - stores integers (whole numbers), without decimals, such as 123 or -123. The `int` data type stores values that are within a range of $-2,147,483,648$ to $2,147,483,647$ (32-bit signed integer). Other useful data types for integers are: `short` (stores 16-bit integers), `long` (stores 32-bit[2] integers), `long long` (stores 64-bit integers).
- **double** - stores floating point numbers, with decimals, such as 19.99 or -19.99. The `double` data type is a double-precision 64-bit floating point. Other useful data types for floating point numbers are: `float` (stores 32-bit floating point numbers), `long double` (stores 128-bit floating point numbers).
- **bool** - stores values with two states: `true` or `false`
- **char** - stores single characters, such as `'a'` or `'B'`. The `char` values are surrounded by single quotes.

  The `char` is a type of integer, so it also stores numbers, which are interpreted as characters according to the ASCII encoding. For example, the ASCII value for the character `'a'` is 97 and the ASCII value for the character `'b'` is 98.
- **std::string** - stores text, such as `"Hello World"`. The `std::string` values are surrounded by double quotes.

[1]The sizes of these data types may vary from one compiler to another.
[2]32-bit on Windows, 64-bits on Linux/MacOS

Excerpts from the C++ library

function	description	example	library
`min()`	returns the minimum of two numbers	`int m = std::min(3, 5);`	`<algorithm>`
`max()`	returns the maximum of two numbers	`std::cout « std::max(3, 5);`	`<algorithm>`
`abs()`	returns the absolute value of a number	`int x = std::abs(-10);`	`<cmath>`
`pow()`	returns the first argument raised to the power of the second argument	`double y = std::pow(2, 3)`	`<cmath>`
`sqrt()`	returns the square root of a number	`double s = std::sqrt(16);`	`<cmath>`
`ceil()`	returns the smallest integer that is greater than or equal to the argument	`int z = std::ceil(9.2);`	`<cmath>`
`floor()`	returns the largest integer that is less than or equal to the argument	`int z = std::floor(9.2);`	`<cmath>`
`lround()`	returns the long integer that is closest to the argument	`long z = std::lround(9.2);`	`<cmath>`
`time()`	returns the current time of the system as time since epoch	`std::cout « std::time(0);`	`<ctime>`
`rand()`	returns a random number	`int r = std::rand();`	`<cstdlib>`
`srand()`	sets the seed for the random number generator	`std::srand(std::time(0));`	`<cstdlib>`
`to_string()`	convert to `std::string` value	`std::string x = std::to_string(17);`	`<string>`
`stoi()`	convert `std::string` value to **`int`** value	`int x = std::stoi("17");`	`<string>`

You need to add appropriate library file to be able to use a function. For instance:

```
#include <iostream>
#include <cmath>
int main() {
    std::cout << "Square roor of 2 is: "
              << std::sqrt(2);
}
```

Lab Questions

1. Develop a program `temperature.cpp` that takes the temperature in Fahrenheit as input and converts it to Celsius using the formula: Celsius = (Fahrenheit - 32.0) * 5.0 / 9.0. Output the converted temperature. Use variables of type `double` for both Fahrenheit and Celsius.

2. Write a program `stats4.cpp` that generates four uniform random integer values, their sum, their average value, and their minimum and maximum values.

   - Use `rand()` and `srand()` from the `<cstdlib>` library to generate the numbers.
   - The average of four integers could be a fractional value, so be sure to use the `double` data type for the average.

   Sample execution:

   ```
   Numbers: 18042 8984 16827 12434
   Sum: 56287
   Average: 14071.8
   Minimum: 8984
   Maximum: 18042
   ```

3. Write a program `three_sort_desc.cpp` that takes three `int` values from the user and prints them in descending (largest to smallest) order.

   - You must not use `if` statements.
   - Use the `min()` and `max()` functions from the `<algorithm>` library to find the largest, smallest, and middle values.

   Sample execution:

   ```
   Enter three values: 101 582 349
   In descending order: 582 349 101
   ```

4. Write a program `wind_chill.cpp` that takes the air temperature and wind speed as input and prints the wind chill index. This index measures how cold people feel when exposed to wind. Your program should take two `double` inputs: `t` (temperature in Fahrenheit) and `v` (wind speed in miles per hour). The output should be the calculated wind chill. Use the following formula from the National Weather Service:

   $$y_0 = 35.74 + 0.6215t - 35.75v^{0.16} + 0.4275tv^{0.16}$$

   - You will need to use the `pow(base, exponent)` function from the `<cmath>` library to calculate $v^{0.16}$.
   - For example, if the temperature is 30°F and the wind speed is 20 mph:

   $$y_0 \approx 35.74 + 0.6215(30) - 35.75(20^{0.16}) + 0.4275(30)(20^{0.16})$$

   $$y_0 \approx 35.74 + 18.645 - 35.75(1.643) + 0.4275(30)(1.643)$$

$$y_0 \approx 35.74 + 18.645 - 58.74 + 21.08 \approx 16.725$$

The answer is approximately 16.7.

Sample execution:

```
Enter temperature (F): 30
Enter wind speed (mph): 20
Wind chill: 16.7289
```

5. The Universal Product Code (UPC) is a 12-digit code used for retail products. The last digit is a check digit calculated as follows:

   - Add all digits in odd positions and multiply the result by 3.
   - Add all digits in even positions to it.
   - The check digit is the number you must add to this sum to make it a multiple of 10.

$$check = (10 - ((oddSum \times 3 + evenSum) \pmod{10})) \pmod{10}$$

Worked example

Given 11 digits: 03600029145 Label positions (left→right): 1:0, 2:3, 3:6, 4:0, 5:0, 6:0, 7:2, 8:9, 9:1, 10:4, 11:5

- Odd positions (1,3,5,7,9,11): $0 + 6 + 0 + 2 + 1 + 5 = 14$ Multiply by 3 $\rightarrow 14 \times 3 = 42$
- Even positions (2,4,6,8,10): $3 + 0 + 0 + 9 + 4 = 16$
- Total: $42 + 16 = 58$
- Check digit: $58 \pmod{10} = 8$ $10 - 8 = 2$ $(10 - 8) \pmod{10} = 2$

So the check digit is 2, and the full UPC is 036000291452.

Input / Output

- Input: one 11-digit integer (you may use `long long` to be safe).
- Output: the 11 digits followed by the computed check digit (12 digits total).

Hint (digit extraction)

If `x` holds the 11-digit number, you can extract digits from the right using % and /, e.g.:

```
int d11 = x % 10;
int d10 = (x / 10) % 10;
int d9  = (x / 100) % 10;
// ...
int d1  = (x / 10000000000LL) % 10; // leftmost
```

Sample run

```
Enter 11-digit number:
3600029145
UPC: 036000291452
```

6. *Day of the week:* Write a program `day_of_week.cpp` that takes a date as input and prints the day of the week that date falls on. Your program should take three inputs: `m` (month), `d` (day), and `y` (year). For `m` use 1 for January, 2 for February, and so forth. For output print 0 for Sunday, 1 for Monday, 2 for Tuesday, and so forth. Use the following formulas, for the Gregorian calendar (where / denotes integer division):

$$\begin{aligned} y_0 &= y - (14 - m)/12 \\ x &= y_0 + y_0/4 - y_0/100 + y_0/400 \\ m_0 &= m + 12 * ((14 - m)/12) - 2 \\ d_0 &= (d + x + 31m_0/12) \pmod 7 \end{aligned}$$

   For example, on which day of the week was August 2, 1953?

$$\begin{aligned} y_0 &= 1953 - 0 = 1953 \\ x &= 1953 + 1953/4 - 1953/100 + 1953/400 = 2426 \\ m_0 &= 8 + 12 * 0 - 2 = 6 \\ d_0 &= (2 + 2426 + (31 * 6)/12) \pmod 7 \\ &= 2443 \pmod 7 = 0 \end{aligned}$$

   The answer is 0, which means Sunday.