**Spring 2023, ISTM, FCU-Purdue 2+2 ECE Program**
**ISTM2572 , Advanced C Programming, Quiz 2**

**Total SEVEN FILES for Quiz 2. Use file name quiz2_DXXXXXXX_1.c for Question 1 and file names quiz2_DXXXXXXX_2.dev, Node_DXXXXXXX.h, Node_DXXXXXXX.cpp, CStack_DXXXXXXX.h, CStack_DXXXXXXX.cpp, and quiz2_DXXXXXXX_2.cpp for Question 2, where DXXXXXXX is your student ID. When you finish a question, submit the above 7 files to the instructor's computer.**

1. (30 points) Start with program skeleton **quiz2_skeleton_1.cpp** and change the file name to **quiz2_DXXXXXXX_1.cpp**. Define and implement a temperature class Celsius. In **class** Celsius, define (1) **double private** data member degree, (2) a default constructor with a **double** parameter of initial value 0.0, (3) a public member function **double** toFahrenheit() **const** converting a Celsius degree to a **double** Fahrenheit degree, and (4) two friend functions for istream and ostream for inputting and outputting a Celsius degree. In the main program, declare a Celsius variable without parameter and write only cout and cin statements to generate the text contents as the following program execution example. Temperature conversion of Celsius degree c to Fahrenheit degree f is: f = c * 9.0 / 5.0 + 32.0.

   Program execution example:

```
Enter a Celsius degree: 100.0
**** Celsius 100 degrees equals to Fahrenheit 212 degrees.
```

   (To be continued)

2. (70 points) Change **Node_skeleton.h**, **Node_skeleton.cpp**, **CStack_skeleton.h**, **CStack_skeleton.cpp**, and **quize1_skeleton_2.cpp** to **Node_DXXXXXXX.h**, **Node_DXXXXXXX.cpp**, **CStack_DXXXXXXX.h**, **CStack_DXXXXXXX.cpp**, and **quiz2_DXXXXXXX_2.cpp**. Create a C++ project **quiz2_DXXXXXXX_2.dev** and add the five .h and .cpp files in the project: **Node_DXXXXXXX.h**, **Node_DXXXXXXX.cpp**, **CStack_DXXXXXXX.h**, **CStack_DXXXXXXX.cpp**, and **quiz2_DXXXXXXX_2.cpp**. Files **Node_DXXXXXXX.h** and **Node_DXXXXXXX.cpp** are the header file and the source file of a single-linked (non-circular) linear list of character (**char**) data elements, files **CStack_DXXXXXXX.h** and **CStack_DXXXXXXX.cpp** are the header file and the source file of character stacks using single-linked linear list implementation, **quiz2_DXXXXXXX_2.cpp** is the source code of application program performing **parentheses matching problem** using character stacks specified and implemented in the problem.

   The character stack **CStack** needs to implement a constructor and four member functions: **void** push(**char**), **char** pop(), **char** top(), and **bool** isEmpty(). Let String str be a sequence of parentheses '()', square brackets '[]', and curly braces '{}'. The **parentheses matching problem** is to check whether str, a string of parentheses of any length, matches or not. For example, "[{((){[]})()}]{()[([])()]}" and "([][[]](){{}}))" are matching parentheses strings; "([][[]]({[{]}}))", "((()()))" are not matching parentheses strings. Repeat parentheses matching test until the input string is "stop".

   Program execution example: (in the next page)

```
Input a string of parentheses ('(', ')', '[', ']', '{', '}'): [{((){[]})()}]{()[([])()]}
**** String [{((){[]})()}]{()[([])()]} is all matching parentheses.

----------------------------------------

Input a string of parentheses ('(', ')', '[', ']', '{', '}'): ([][[]](){{}}))
**** String ([][[]](){{}})) is all matching parentheses.

----------------------------------------

Input a string of parentheses ('(', ')', '[', ']', '{', '}'): ([][[]]({[{]}}))
**** String ([][[]]({[{]}})) is not all matching parentheses.

----------------------------------------

Input a string of parentheses ('(', ')', '[', ']', '{', '}'): ((()()))
**** String ((()())) is not all matching parentheses.

----------------------------------------

Input a string of parentheses ('(', ')', '[', ']', '{', '}'): stop
```