BITP 1113: PROGRAMMING TECHNIQUES
1BITS
Lab 6
Function Part 1

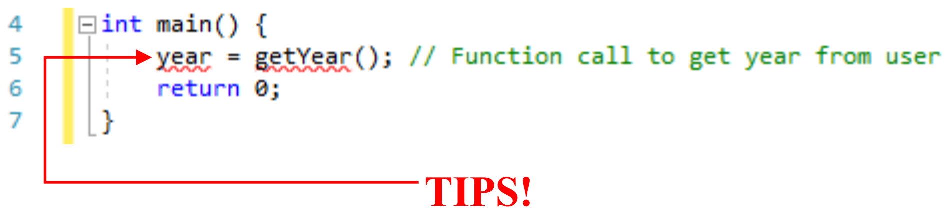
Exercise 1

A company keeps a chart of the number of working days in a year. Employees have Saturday and Sunday off which Saturday and Sunday are 52 days each in a year. Write a program to compute and display a number of working days in a year. User will provide the year which could be a leap year.

1. Analyse input, process and output.

   Input : `year`
   Process : if leap year, `workingDays =  366 - 52 * 2;`
   `Else, workingDays = 365 - 52 * 2;`
   Output : `year, workingDays`

2. We will define three programmer-defined functions as follows:
   a. Function to get a year from user.
   b. Function to determine whether the year is a leap year.
   c. Function to compute the working days.
3. Create a new project named `Days`
4. At **Solution Explorer**, right-click **Source Files** folder and add new item named `WorkingDays.cpp`.
5. Write the default code.
6. Write function call for `getYear()` to call a function to get a `year` from a user (Line 5).



```
4  int main() {
5      year = getYear(); // Function call to get year from user
6      return 0;
7  }
```

TIPS!

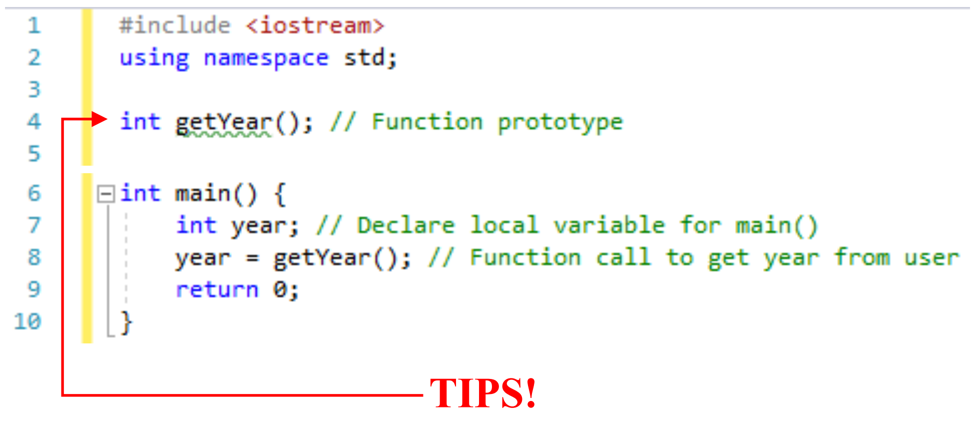
1. What type of function could be used in our program?

TIPS: Considering the RECEIVE and RETURN values.

The called function, i.e. `getYear()` does not need any value from a calling function, i.e. `main()` to perform its task, so there is no passing variable inside the parentheses.
The entered year will be used by the calling function, i.e. `main()` function. So there must be a received variable at function call to hold the returned year value.

2. To avoid syntax errors, declare the variable as usual and declare function getYear() using function prototype.

7. Declare local variables `year` for `main()` (Line 7) and write a function prototype for `getYear()` (Line 4).

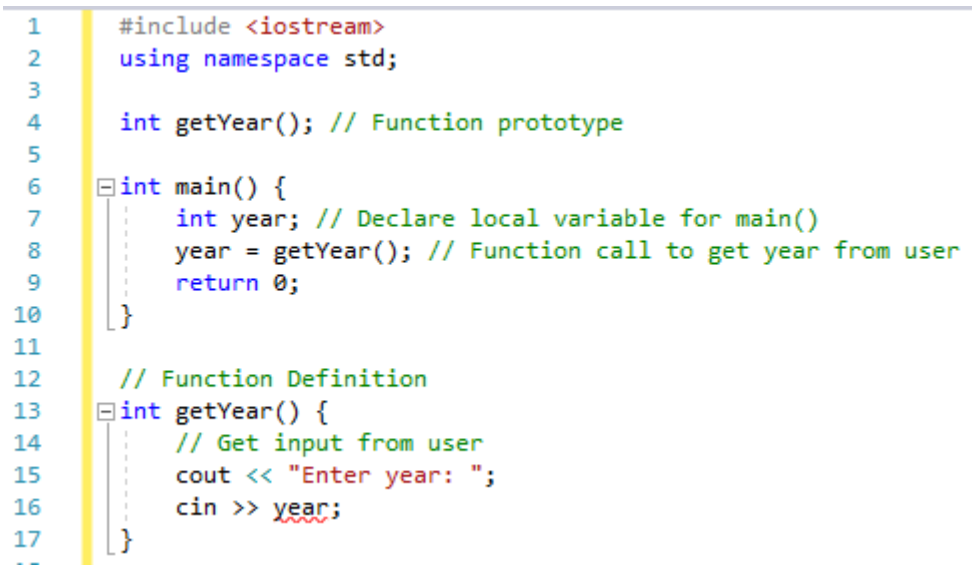


```
1   #include <iostream>
2   using namespace std;
3
4   int getYear(); // Function prototype
5
6   int main() {
7       int year; // Declare local variable for main()
8       year = getYear(); // Function call to get year from user
9       return 0;
10  }
```

TIPS!

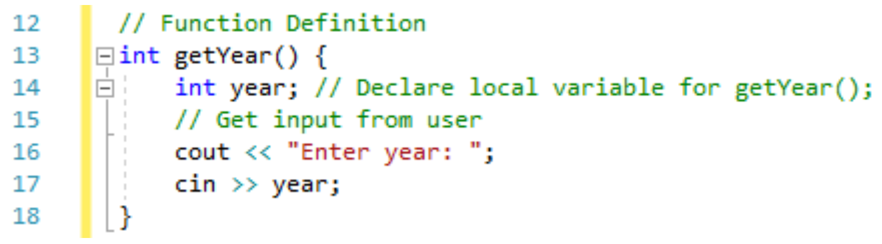
The function name with green underline indicates the function definition is missing.

8. Write the function definition for `getYear()`. Insert code to receive a `year` from a user (Line 12 to 17).



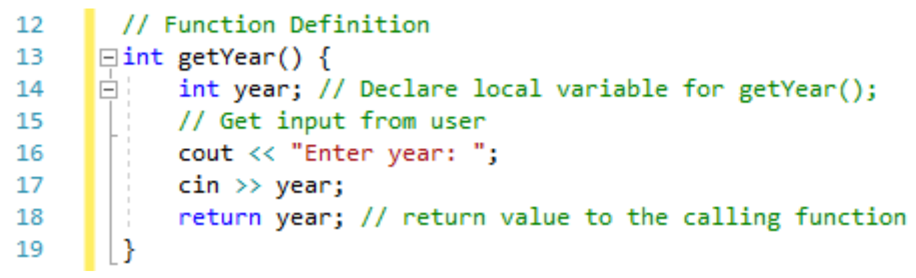
```
1   #include <iostream>
2   using namespace std;
3
4   int getYear(); // Function prototype
5
6   int main() {
7       int year; // Declare local variable for main()
8       year = getYear(); // Function call to get year from user
9       return 0;
10  }
11
12  // Function Definition
13  int getYear() {
14      // Get input from user
15      cout << "Enter year: ";
16      cin >> year;
17  }
```

9. Declare the local variable `year` for `getYear()` (Line 14).



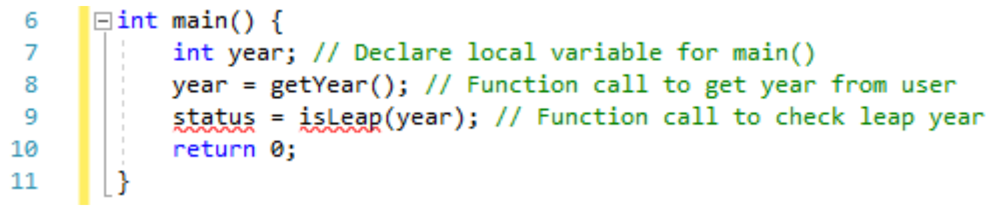
```
12  // Function Definition
13  int getYear() {
14      int year; // Declare local variable for getYear();
15      // Get input from user
16      cout << "Enter year: ";
17      cin >> year;
18  }
```

10. Insert `return` statement at the end of `getYear()` function definition to return `year` to the calling function, i.e. `main()` (Line 18).



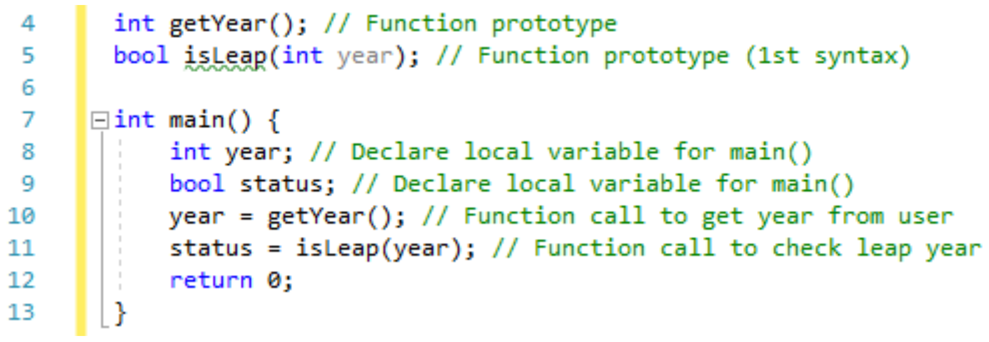
```
12    // Function Definition
13   int getYear() {
14       int year; // Declare local variable for getYear();
15       // Get input from user
16       cout << "Enter year: ";
17       cin >> year;
18       return year; // return value to the calling function
19   }
```

11. Write function call for `isLeap()` to call a function to check whether the `year` entered is a leap year. This function should return the result because the working days calculation depends on this result (Line 9).



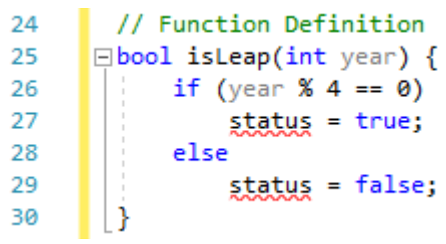
```
6    int main() {
7        int year; // Declare local variable for main()
8        year = getYear(); // Function call to get year from user
9        status = isLeap(year); // Function call to check leap year
10       return 0;
11   }
```

12. Declare local variables `status` for `main()` (Line 9) and write a function prototype for `isLeap()` (Line 5).



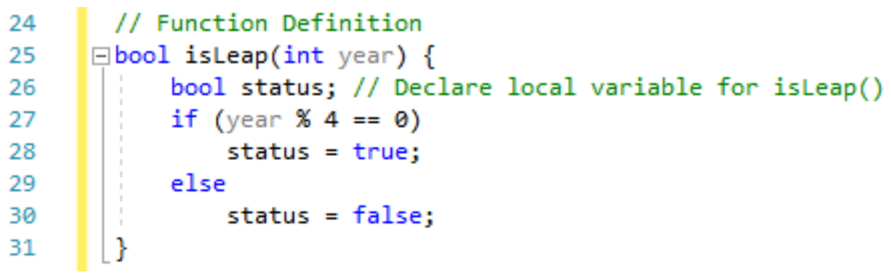
```
4    int getYear(); // Function prototype
5    bool isLeap(int year); // Function prototype (1st syntax)
6
7    int main() {
8        int year; // Declare local variable for main()
9        bool status; // Declare local variable for main()
10       year = getYear(); // Function call to get year from user
11       status = isLeap(year); // Function call to check leap year
12       return 0;
13   }
```

13. Write the function definition for `isLeap()`. Insert code to check the `year` status (Line 24 to 30).



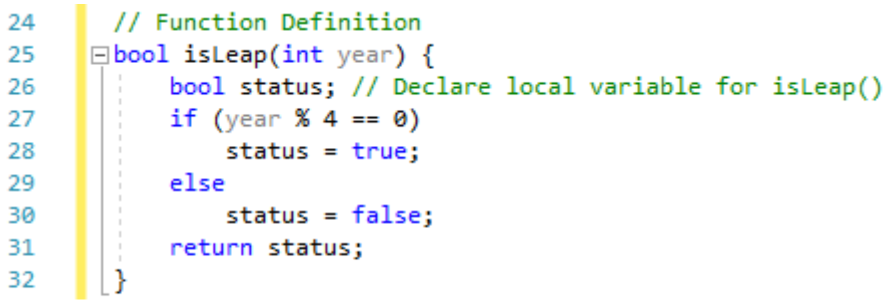
```
24    // Function Definition
25   bool isLeap(int year) {
26       if (year % 4 == 0)
27           status = true;
28       else
29           status = false;
30   }
```

14. Declare the local variable `status` for `isLeap()` (Line 26).



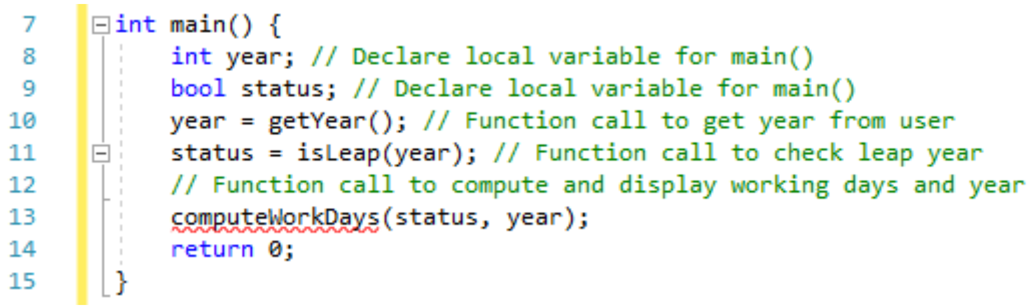
```
24    // Function Definition
25   bool isLeap(int year) {
26       bool status; // Declare local variable for isLeap()
27       if (year % 4 == 0)
28           status = true;
29       else
30           status = false;
31   }
```

15. Insert `return` statement at the end of `isLeap()` function definition to return `status` to the calling function, i.e. `main()` (Line 31).



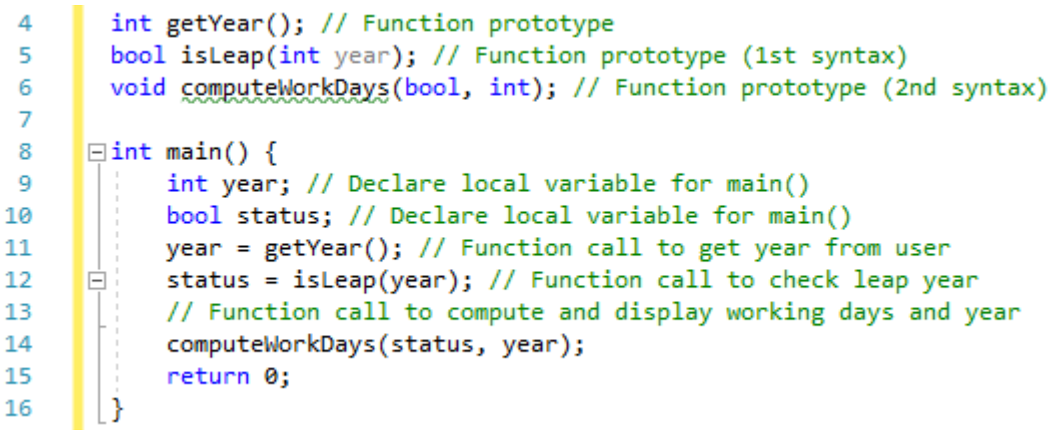
```
24    // Function Definition
25   bool isLeap(int year) {
26       bool status; // Declare local variable for isLeap()
27       if (year % 4 == 0)
28           status = true;
29       else
30           status = false;
31       return status;
32   }
```

16. Write function call for `computeWorkingDays()` to call a function to calculate the number of working days. This function will not return the result because the working days will be displayed in this function (Line 13).



```
 7   int main() {
 8       int year; // Declare local variable for main()
 9       bool status; // Declare local variable for main()
10       year = getYear(); // Function call to get year from user
11       status = isLeap(year); // Function call to check leap year
12       // Function call to compute and display working days and year
13       computeWorkDays(status, year);
14       return 0;
15   }
```

17. Write a function prototype for `computeWorkingDays()` (Line 6).



```
 4   int getYear(); // Function prototype
 5   bool isLeap(int year); // Function prototype (1st syntax)
 6   void computeWorkDays(bool, int); // Function prototype (2nd syntax)
 7
 8   int main() {
 9       int year; // Declare local variable for main()
10       bool status; // Declare local variable for main()
11       year = getYear(); // Function call to get year from user
12       status = isLeap(year); // Function call to check leap year
13       // Function call to compute and display working days and year
14       computeWorkDays(status, year);
15       return 0;
16   }
```

18. Write the function definition for `computeWorkingDays()`. Insert code to calculate and display the `year` and working days (Line 38 to 46).

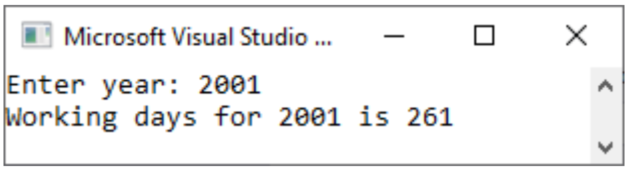
```
38    // Function Definition
39  void computeWorkDays(bool status, int year) {
40        cout << "Working days for " << year << " is ";
41        if (status == true)
42            cout << 366 - 52 * 2;
43        else
44            cout << 365 - 52 * 2;
45        cout << endl;
46  }
```

19. Insert `return` statement without return variable at the end of `computeWorkingDays()` (Line 46).

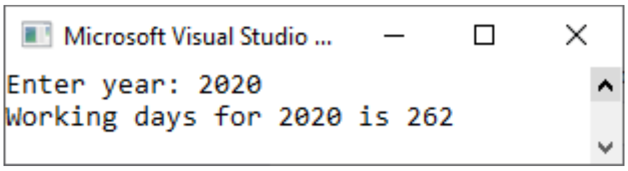
```
38    // Function Definition
39  void computeWorkDays(bool status, int year) {
40        cout << "Working days for " << year << " is ";
41        if (status == true)
42            cout << 366 - 52 * 2;
43        else
44            cout << 365 - 52 * 2;
45        cout << endl;
46        return;
47  }
```

20. Compile and run to observe the output.

Example of Output



```
Enter year: 2001
Working days for 2001 is 261
```



```
Enter year: 2020
Working days for 2020 is 262
```

Lab Attendance Week 6 (Group)
Question 1

a) What types of function are available in this program.

b) Based on the program, state the functions that match with your answer at Question (a).

Question 2

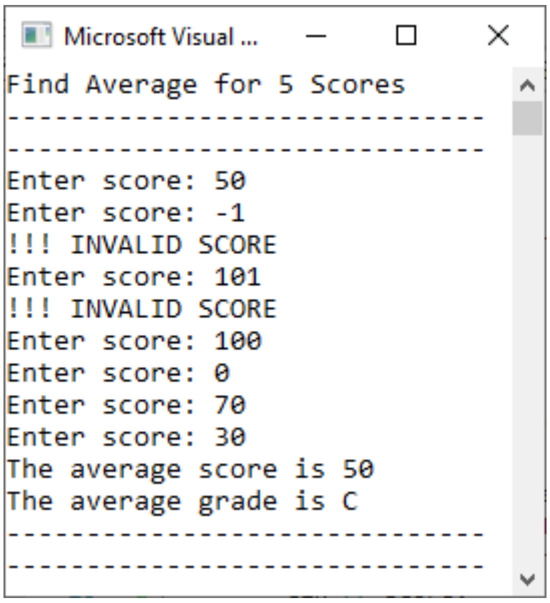
Write a program to find the average grade based on Table 1 provided. The user will enter five scores, then the program will display the average score and the grade.

Notes: You must apply ALL four types of function in your program.

Table 1: Average Score-Grade Table

Average Score	Grade
80 to 100	A
65 to 79	B
50 to 64	C
40 to 49	D
0 to 39	F

Example Output



```
Find Average for 5 Scores
------------------------------
------------------------------
Enter score: 50
Enter score: -1
!!! INVALID SCORE
Enter score: 101
!!! INVALID SCORE
Enter score: 100
Enter score: 0
Enter score: 70
Enter score: 30
The average score is 50
The average grade is C
------------------------------
------------------------------
```

Submit this exercise at ULearn before 12.00 p.m. 10 December 2020 (Thursday).

~Push yourself because no one else is going to do it for you~
- Anonymous