UEE 1303(1069): Object-Oriented Programming

Lab #1: Advances Topic on Functions

In this laboratory session you will:

- Learn how to use the inline function, function overloading and function template

Lab 1-1: Inline Functions

✓ The following two examples are used to demonstrate the difference between `inline` functions and `define` macro.

```
// lab1-1-1.cpp
#include <iostream>
#define PI 3.14159
#define circleArea(x) (PI*(x)*(x))

int main()
{
    std::cout << "circleArea(4.3) = "
              << circleArea(4.3) << std::endl;
    std::cout << "circleArea(4) = "
              << circleArea(4) << std::endl;

    return 0;
}
```

```
// lab1-1-2.cpp
#include <iostream>
#define PI 3.14159
inline double circleArea(int x) {return PI*x*x;}

int main()
{
    std::cout << "circleArea(4.3) = "
              << circleArea(4.3) << std::endl;
    std::cout << "circleArea(4) = "
              << circleArea(4) << std::endl;
```

```
    return 0;
}
```

- Although `define` macro results in the answer as you expect, `inline` functions follow all the protocols of type safety enforced on normal functions. When you compile the `lab1-1-2.cpp` file with `inline` functions, your compiler will show the warning message about erroneous type conversion.

Lab 1-2: FUNCTION OVERLOADING

✓ Execute the following program lab1-2

```
//File: lab1-2.cpp
#include <iostream>
using namespace std;
double avg(double, double, double);
double avg(double, double);
int main()
{
   cout << "The average of 2.0, 2.5, and 3.0 is "
        << avg(2.0, 2.5, 3.0) << endl;
   cout << "The average of 4.5 and 5.5 is "
        << avg(4.5, 5.5) << endl;
   return 0;
}
double avg(double n1, double n2, double n3)
{
   return ((n1 + n2 + n3) / 3.0);
}
double avg(double n1, double n2)
{
   return ((n1 + n2) / 2.0);
}
```

✓ Modify the program lab1-2.cpp as lab1-2-1.cpp

```
//File: lab1-2-1.cpp
#include <iostream>
#include <cmath>
using namespace std;
```

```
int avg(double, double);
double avg(double, double);
int main()
{
    cout << "The average of 2.0 and 3.0 is "
         << avg(2.0, 3.0) << endl;
    cout << "The average of 2.0 and 3.0 is "
         << avg(2.0, 3.0) << endl;
    return 0;
}
int avg(double n1, double n2)
{
    return (int) ceil((n1 + n2) / 2.0);
}
double avg(double n1, double n2)
{
    return ((n1 + n2) / 2.0);
}
```

- ✓ Note that the functions can only be overloaded by different arguments but cannot be overloaded by different return types.
- ✓ `ceil` is a round up value which is used to return the smallest integral value that is not less than x.
- ✓ Please modify the program to obtain the correct results.

Lab 1-3: FUNCTION TEMPLATE

- ✓ Execute the following program lab1-3, you should edit two files: lab1-3.cpp and maximum.h

```
// lab1-3.cpp
#include <iostream>
#include "maximum.h"
using namespace std;

int main()
{
    int int1, int2, int3;
    cout << "Input three integers: ";
    cin >> int1 >> int2 >> int3;
```

```
    cout << "Maximum is " << maximum(int1, int2, int3);

    double double1, double2, double3 ;
    cout << "Input three double variables: ";
    cin >> double1 >> double2 >> double3;
    cout << "Maximum is " << maximum(double1, double2, double3);

    char char1, char2, char3 ;
    cout << "Input three characters: ";
    cin >> char1 >> char2 >> char3;
    cout << "Maximum is " << maximum(char1, char2, char3);

    return 0;
}
```

```
//  maximum.h
template <class T>
T maximum(T value1, T value2, T value3)
{
    T max = value1;

    if (value2 > max)
        max = value2;
    if (value3 > max)
        max = value3;

    return max;
}
```

Exercise 1-1

✓ Write a C++ program that prompts the user for the radius of a circle, then calls inline function circleArea to calculate the area of that circle

```
>./ex1-1
Enter the radius of the circle: 4
The area of the circle is 50.2654
>
```

Exercise 1-2

1. Write a function template **`selectionSort`** based on *sample.cpp*. Write a driver program that inputs, sorts and outputs an **`int`** array and a **`float`** array.

   The **`int`** array should be

   ```
   int a[ SIZE ] = { 2, 9, 10, 1, 7, 3, 4, 5, 8, 6 };
   ```

   The **`float`** array should be

   ```
   double b[ SIZE ] = { 2.2, 9.9, 10.1, 1.1, 7.7, 3.3, 4.4,
   5.5, 8.8, 6.6};
   ```

```
>./ex1-2
int data items in original order
 2       9      10       1       7       3       4       5       8       6
int data items in ascending order
1       2       3       4       5       6       7       8       9      10
float data items in original order
2.2     9.9   10.1     1.1     7.7     3.3     4.4     5.5     8.8     6.6
float point data items in ascending order
1.1     2.2     3.3     4.4     5.5     6.6     7.7     8.8     9.9   10.1
>
```