Week 5 Lab

Functions

Functions are generally used to calculate a value and return that value (result). The following program calculate the area of a rectangle (length*width).

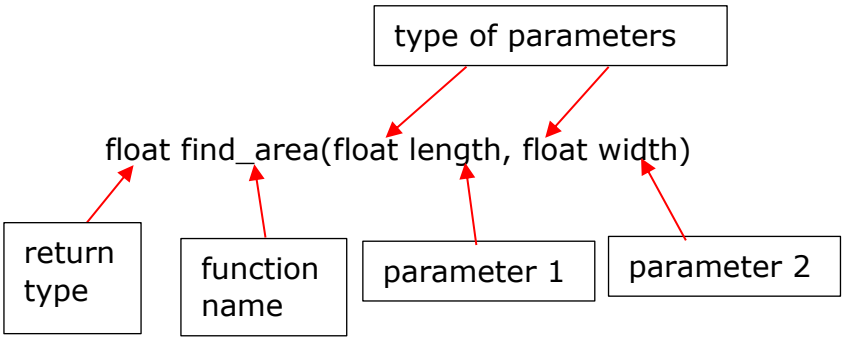
```
#include <iostream>
using namespace std;
int main()
{
    float length,width, rect_area;
    cout<<"Enter length and width of the rectangle: \n";
    cin>>length>>width;

    rect_area = length * width;

    cout<<Area of the rectangle is "<< rect_area <<endl;
    return 0;
}
```

Instead of doing the calculation in main you can write a separate function that will return the result (in this case area) whenever it is called from the main function.

You need to define a function, and this function needs two values **length** (float value) and **width** (float value) to calculate the area. So when you call this function it is your responsibility to provide these values (known as **parameters** or **arguments**) to the function. This function will do the calculation of the area and the result will be returned. Here the area (result) will be a fractional value; therefore the return value will be float or double. Now you can write the function prototype.



The body of the function will calculate the area with the parameters (length and width) provided and the result (area) will be returned to the caller when the function ends.

```
float find_area(float length, float width)
{
      float area;

      area = length * width;
      return area;
}
```

Now it is possible to call the **find_area** function in main, and store the return result of calling the function into main's **rect_area** variable.

```
int main()
{
    float length,width,rect_area;
    cout<<"Enter length and width of the rectangle: \n";
    cin>>length >> width;

    rect_area = find_area(length,width);

    cout<<"Area of the rectangle is "<< rect_area;
    return 0;
}
```

The complete program will look like this. Please remember to write your function before main function.

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

float find_area(float length, float width)
{
    float area;

    area = length * width;
    return area;
}

int main()
{
    float length,width,rect_area;
    cout<<"Enter length and width of the rectangle: \n";
    cin>> length >> width;

    rect_area = find_area(length,width);

    cout<<"Area of the rectangle is " << fixed <<setprecision(2) << rect_area;
    return 0;
}
```

Task 5.1 – Find the error

a)

```
int g()
{
      cout << "Inside function g" << endl;
      int h()
      {
            cout << "Inside function h" << endl;
      }
}
```

b)

```
int sum( int x, int y )
{
      int result;
      result = x + y;
}
```

c)

```
int sum( int n )
{
      if ( n == 0 )
      return 0;
      else
      n + sum( n - 1 );
}
```

d)

```
void f( double a );
{
      float a;
      cout << a << endl;
}
```

e)

```
void product()
{
      int a;
      int b;
      int c;
      int result;
      cout << "Enter three integers: ";
      cin >> a >> b >> c;
```

```
        result = a * b * c;
        cout << "Result is " << result;
        return result;
    }
```

f)

```
    template < class A >
    int sum( int num1, int num2, int num3 )
    {
        return num1 + num2 + num3;
    }
```

g)

```
    void printResults( int x, int y )
    {
        cout << "The sum is " << x + y << '\n';
        return x + y;
    }
```

h)

```
    template < A >
    A product( A num1, A num2, A num3 )
    {
        return num1 * num2 * num3;
    }
```

i)

```
    double cube( int );
    int cube( int );
```

Task 5.2 Give the function header for each of the following functions:

a) Function hypotenuse that takes two double-precision floating-point arguments side1 and side2, and returns a double-precision floating-point result.

b) Function smallest that takes three integers x, y and z, and returns an integer.

c) Function instructions that does not receive any arguments and does not return a value. [Note: Such functions are commonly used to display instructions to a user.]

d) Function intToDouble that takes an integer argument number, and returns a doubleprecision floating-point result.

(Need to submit Task 5.3 or Task 5.4 as a part of the assignment 2)

Task 5.3

Write a program that plays the game of "guess the number" as follows: Your program chooses the number to be guessed by selecting an integer at random in the range 1 to 1000. The program then displays the following:

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
```

The player then types a first guess. The program responds with one of the following:

```
1. Excellent! You guessed the number!
   Would you like to play again (y or n)?
2. Too low. Try again.
3. Too high. Try again.
```

If the player's guess is incorrect, your program should loop until the player finally gets the number right. Your program should keep telling the player "Too high" or "Too low" to help the player "zero in" on the correct answer.

Modify this program to count the number of guesses the player makes. If the number is 10 or fewer, print "Either you know the secret or you got lucky!" If the player guesses the number in 10 tries, then print "Ahah! You know the secret!" If the player makes more than 10 guesses, then print "You should be able to do better!"

Write at least two separate functions other than main for this exercise.

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too high. Try again.
? 250
Too low. Try again.
? 375
Too high. Try again.
? 313
Too low. Try again.
? 344
Too low. Try again.
? 360
Too high. Try again.
? 352
Too high. Try again.
? 348
Too low. Try again.
? 350
Too low. Try again.
? 351

Excellent! You guessed the number!
Ahah! You know the secret!
Would you like to play again?
Please type (y/n)? n
Press any key to continue
```

Task 5.4

A player rolls two dice. Each die has six faces. These faces contain 1, 2, 3, 4, 5 and 6 spots. After the dice have come to rest, the sum of the spots on the two upward faces is calculated. If the sum is 7 or 11 on the first roll, the player wins. If the sum is 2, 3 or 12 on the first roll, the player loses. If the sum is 4, 5, 6, 8, 9 or 10 on the first roll, then that sum becomes the player's "point." To win, you must continue rolling the dice until you "make your point." The player loses by rolling a 7 before making the point.

Then modify your program to allow wagering. Initialize variable *bankBalance* to 100 dollars. Prompt the player to enter a wager. Use a while loop to check that wager is less than or equal to *bankBalance* and, if not, prompt the user to re-enter wager until a valid wager is entered. After a correct wager is entered, run game. If the player wins, increase bankBalance by wager and print the new *bankBalance*. If the player loses, decrease *bankBalance* by wager, print the new *bankBalance*, check on whether *bankBalance* has become zero and, if so, print the message "Sorry. You busted!" As the game progresses, print various messages to create some "chatter" such as "Oh, you're going for broke, huh?", "Aw cmon, take a chance!" or "You're up big. Now's the time to cash in your chips!"

Write at least two separate functions other than main for this exercise.

```
You have $1000 in the bank.
Place your wager: 1000
Player rolled 5 + 4 = 9
Point is 9
Way too lucky! Those dice have to be loaded!
You're up big. Now's the time to cash in your chips!
Player rolled 5 + 4 = 9
Player wins
Your new bank balance is $2000
Would you like to try your luck again (y/n)? n
```