Exercise solution

Function Call and Return

- What is the output?

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

int fun(int x = 0, int y = 0, int z = 9)
{  return (x + y + z); }

int main()
{
   cout << fun(10);
   return 0;
}
```

output: 19

Function Call and Return

- What is the output?

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

int fun(int x = 0, int y = 0, int z = 9)
{  return (x + y + z); }

int main()
{
   cout << fun(10);
   return 0;
}
```

output: 19

Function Call and Return

- What is the output?

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

int fun(int x = 0, int y = 0, int z = 9)
{  return (x + y + z); }

int main()
{
   cout << fun(10,0,0);
   return 0;
}
```

output: 10

Function Call and Return

- What is the output?

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

int fun(int x = 0, int y = 0, int z = 9)
{  return (x + y + z); }

int main()
{
   cout << fun(10,0,-3);
   return 0;
}
```

output: 7

Function Call and Return

- What is the output?

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

int fun(int x = 0, int y = 0, int z = 9)
{  return (x + y + z); }

int main()
{
   cout << fun(10,0);
   return 0;
}
```

output: 19

Function Call and Return

- Write a Boolean function which takes a single integer parameter, and returns "True" if the integer is a prime number between 1 and 1000.

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

int get_valid_number();
bool test_prime(int integer);
int main()
{
  int number;
  cout << "This program tests to see if an integer\n is a prime between 1 and
1000.\n\n\n";
cout << "Enter a number (0 to exit)\n";
  number = get_valid_number();
   while (number != 0) {
    cout << "The number " << number << " is ";
    if (!test_prime(number)  )
      cout << "not ";
    cout << "a prime between 1 and 1000.\n\n";
    cout << "Enter a number (0 to exit)\n";
    number = get_valid_number();
  }
 return 0;
}
```

Write the body of :

- `get_valid_number`
- `test_prime`

```
int get_valid_number() {
  whie(number > 1 && number <1000)
    cin>>number;
  return number;
  }
```

```
bool test_prime(int x) {
  if(x %2 == 0)
    return true;
  else
    return false;
  }
```

Function Call and Return

- Write a double function that evaluates a quadratic equation of the form: $ax^2 + bx + c$

Write the body of :
- Quadratic
- checkResponse

```
#include <iostream>
#include<cmath>
using namespace std;
double Quadratic(double, double, double, double);
bool checkResponse(void);
int main(){
double a, b, c;    //coefficients of ax2 + bx + c
double value_x;      //value at which the quadratic will be evaluated
double answer;     //value of the quadratic equation
bool keepGoing;   //Does the user want to continue?
keepGoing = true;
//as long as the user wants to process quadratic equations do this loop
while (keepGoing)
{//Read in values a, b, c, and x
cout << "Please enter coefficients for the quadratic" <<" equation" << endl;
cin >> a >> b >> c;
cout << "Please enter the value at which the quadratic should be evaluated" << endl;
cin >> value_x;
//Evaluate the quadratic
answer = Quadratic(a,b,c, value_x)  ;
//Display the results
cout << endl << "The quadratic equation " << a << "x^2 + " << b << "x + " << c << " has a
function value of " << answer << endl << "when x = " << value_x << "." << endl;
keepGoing = checkResponse()  ;}
return 0;
}
```

```
double Quadratic(double a, double b, double c, double x) {
   int answer;
   answer= a*pow(x,2) + b*x + c ;
   return answer;
}

bool checkResponse(void) {
  char ans;
  bool flag = true ;
  cout<< " do you want to continue? (press y for yes , n for  no)";
   Cin>>ans;
   if( ans == 'y' II ans == 'Y' )
     flag = true;
   if (ans == 'n' II ans == 'N')
     flag = false;
   return flag ;
}
```

Variables Scopes

```
include <iostream>
using namespace std;


int main()
{
int count = 1;
  for (; count <= 5 ; count++)
  {
        int count = 4;
        cout << count << "\n";
  }
return 0;}
```

Output:

```
4
4
4
4
4
```

Variables Scopes

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

int main()
{
int count = 1;
  while (count <= 5)
  {
        int count = 4;
        cout << count << "\n";
        count++;
  }
return 0;}
```

Output:

if we want to increase the count in outside while loop, we will increse it before int count = 4

the Output will be cout infinte number of "4" becouse we increase the count that inside the while loop not the outside

Global Vs. Local Variables

- What is the output?

```
#include <iostream>
using namespace std;
// Global variable declaration:
int g = 20;
int main ()
{ // Local variable declaration:
int g = 10;
cout << g;
return 0; }
```

Output: 10

Global Vs. Local Variables

- What is the output?

```
#include <iostream>
using namespace std;
void func1(){
      int x = 4; // local to func1
      cout << x << endl;}

void func2(){
      int x = 5; // local to func2
      cout << x << endl;}

int main(){
      func1();
      func2();
      return 0;}
```

Output:

```
4
5
```

Global Vs. Local Variables

- What is the output?

```
#include <iostream>
using namespace std;
int g = 10; // global variable

void func1(){
      g = 20; // changing the global g
      cout << g << endl;
}

int main(){
      func1();
      cout << g << endl;
     g = 30; // changing the global g
      return 0;}
```

Output:

```
20
20
```

Global Vs. Local Variables

- What is the output?

```
#include <iostream>
using namespace std;
// Global variable declaration:
int g = 20;
int main ()
{ // Local variable declaration:
int g = 10;
cout << g;
return 0; }
```

Try :

```
int g=10;
cout<< g;10
cout<< ::g; 20
```

Global Vs. Local Variables Questions

- What happened if you redefined global variable in the main ?
  - Does it produce an error ?
  - Why ? No, because of its in a different scope
- if you use the global variable in the main function by assigning new value, what is the final result holds in the global variable?
- What is the effect of using : : before any global variables?

2- the global variable will be chage.

3- its will print the Global variable NOT the Local variable

Call by Value Vs. Call by Reference

```
#include <iostream>
using namespace std;
void swap(int &x, int &y);
int main () {
   // local variable declaration:
   int a = 100;
   int b = 200;
   cout << "Before swap, value of a :" << a << endl;
   cout << "Before swap, value of b :" << b << endl;
   swap(a, b);
   cout << "After swap, value of a :" << a << endl;
   cout << "After swap, value of b :" << b << endl;
   return 0;}
void swap(int &x, int &y) {
   int temp;
   temp = x; x = y;  y = temp;
  return;
}
```

Change the function to be call by value function

```
Before swap, value of a :100
Before swap, value of b : 200
after swap, value of a: 200
after swap, value of b: 100
```

Call by Value Vs. Call by Reference Questions

- Answer the following questions:
  - Passing an entire array to function as parameter is considered passing by reference or passing value? passing by reference
  - What if you pass some elements? Is it considered as passing by reference or value? passing by value
  - In case you want to pass an array without giving the authority to change it. What can you do in order to protect it?

    declere the array as a constant
    like:
    int array(const int ar[ ], int size);