Homework 2 Report

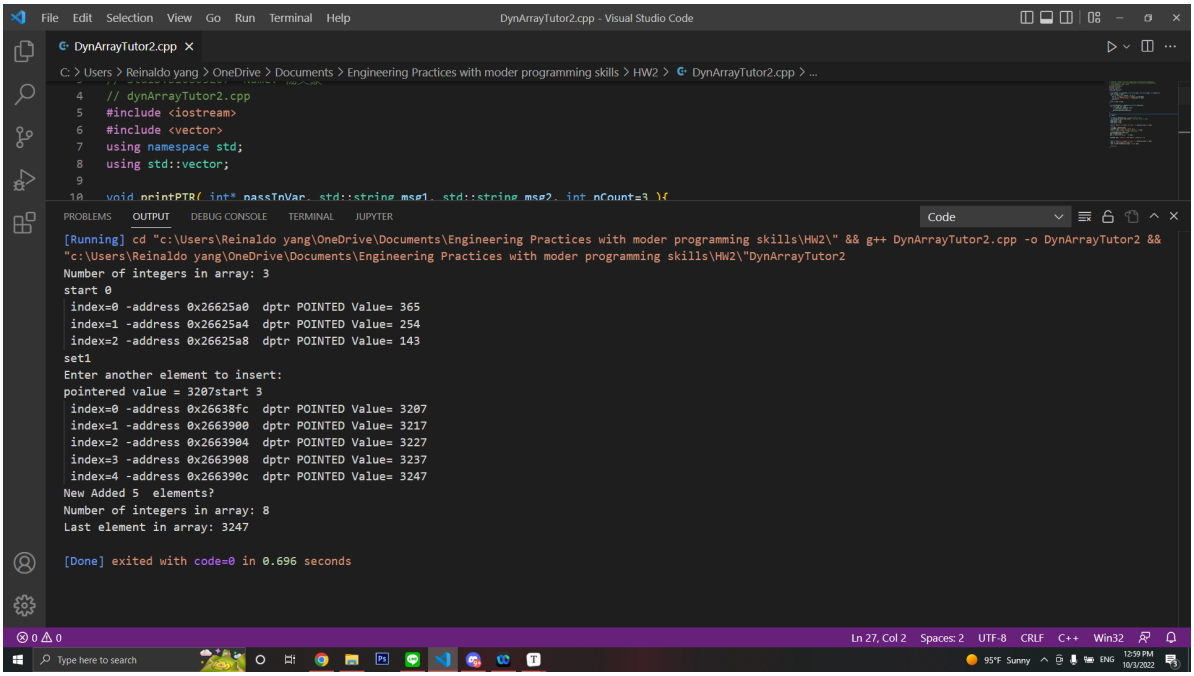
Name	Student ID	Grades
楊英豪	B10803207	

Code Dynamic Array

```
// Creating a dynamic array of integers and inserting values dynamically
// Test this at C++ tutor: https://pythontutor.com/visualize.html#mode=edit
// StuID:B10803207  Name: 楊英豪
// dynArrayTutor2.cpp
#include <iostream>
#include <vector>
using namespace std;
using std::vector;

void printPTR( int* passInVar, std::string msg1, std::string msg2, int nCount=3 )
{
  cout << msg1 << endl;
  for ( int ix=0 ; ix < nCount ; ++ ix) {
    cout <<  " index=" << ix << " -address "<< passInVar
     << "\t dptr POINTED Value= " << *passInVar << endl;
    passInVar++;
  }
  cout << msg2 << endl;
}

void insertElement(int newValue,vector<int>& dynArray){
  for ( int i=0; i<5; ++i){
      int newValue2 = newValue+ i*10;
      // cout<<newValue<<endl;
      dynArray.push_back(newValue2);
    }
}

int main()
{
  vector<int> dynArray (3); // dynamic array of int
  int n5Array[5] ={1111,2222,3333,3207}; // add your StuID as Num#4
  dynArray[0] = 365;
  dynArray[1] = 254;
  dynArray[2] = 143;

  cout << "Number of integers in array: " << dynArray.size() << endl;
  // set1
  int* dptr = &dynArray[0];
  printPTR( dptr, "start 0", "set1", 3 );
  cout << "Enter another element to insert: " << endl;
```

```
  // int newValue = 2220; // TODO: your stuID
  insertElement(3207,dynArray);
 dptr = &dynArray[3];
 cout << "pointered value = " << *dptr;

 printPTR( dptr, "start 3", "New Added 5  elements?", 5 );


  cout << "Number of integers in array: " << dynArray.size() << endl;
  cout << "Last element in array: ";
  cout << dynArray[dynArray.size() - 1] << endl;

  return 0;
}
```

Output:

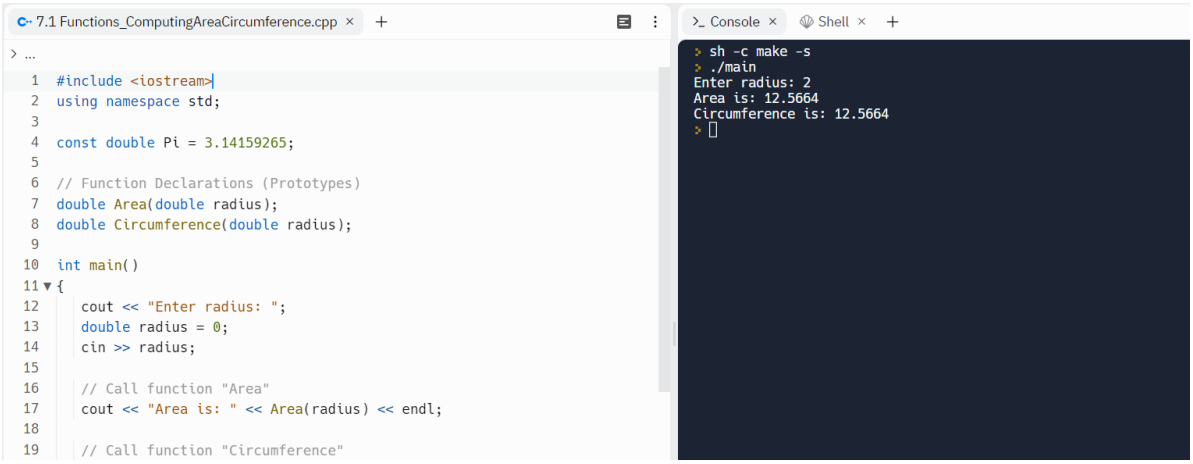


Example 7.1

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

const double Pi = 3.14159265;

// Function Declarations (Prototypes)
double Area(double radius);
double Circumference(double radius);

int main()
{
   cout << "Enter radius: ";
   double radius = 0;
   cin >> radius;
```

```
    // Call function "Area"
    cout << "Area is: " << Area(radius) << endl;

    // Call function "Circumference"
    cout << "Circumference is: " << Circumference(radius) << endl;

    return 0;
}

// Function definitions (implementations)
double Area(double radius)
{
    return Pi * radius * radius;
}

double Circumference(double radius)
{
    return 2 * Pi * radius;
}
```

Output



心得

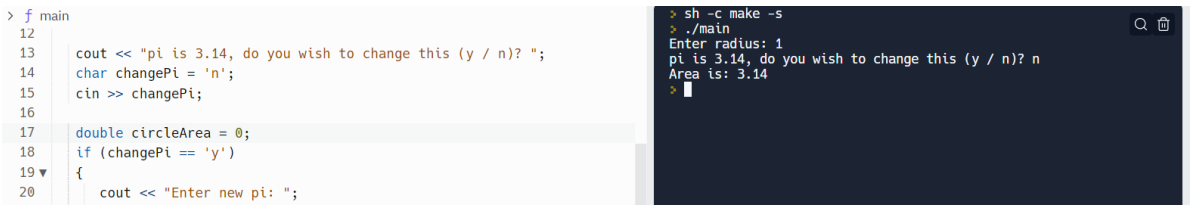
A recommended way to make code readable for other programmer that takes a look of your code is to have a function prototype, it prevents other programmer to be bombarded with all of the details of the function, instead will be shown the return value type, function name, and also the function parameters. We usually put function prototype before the main function so that on reaching line 16 and 19 the compiler will know that area **Area** and **Circumference** are functions. After the main function there will be the function definition, which will be the function body, which will be executed using function call in the main body.

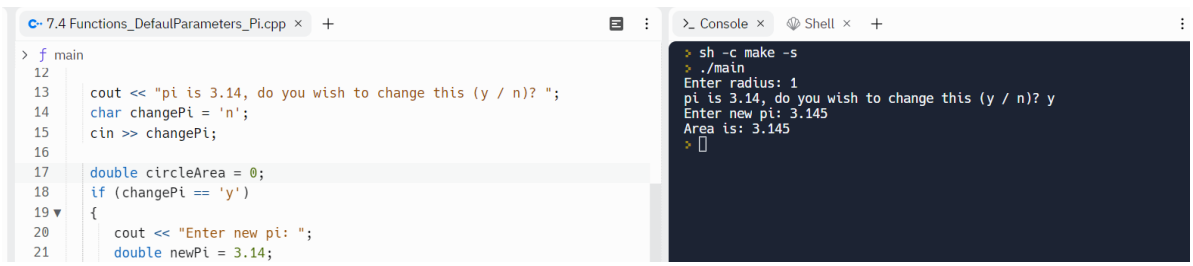
Example 7.4

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

// Function declaration with default argument
double Area(double radius, double pi = 3.14);
```

```
int main()
{
   cout << "Enter radius: ";
   double radius = 0;
   cin >> radius;

   cout << "pi is 3.14, do you wish to change this (y / n)? ";
   char changePi = 'n';
   cin >> changePi;

   double circleArea = 0;
   if (changePi == 'y')
   {
      cout << "Enter new pi: ";
      double newPi = 3.14;
      cin >> newPi;
      circleArea = Area (radius, newPi);
   }
   else
      circleArea = Area(radius); // Ignore 2nd param, use default value

   // Call function "Area"
   cout << "Area is: " << circleArea << endl;

   return 0;
}

// Function definitions (implementations)
double Area(double radius, double pi)
{
   return pi * radius * radius;
}
```

Output





心得

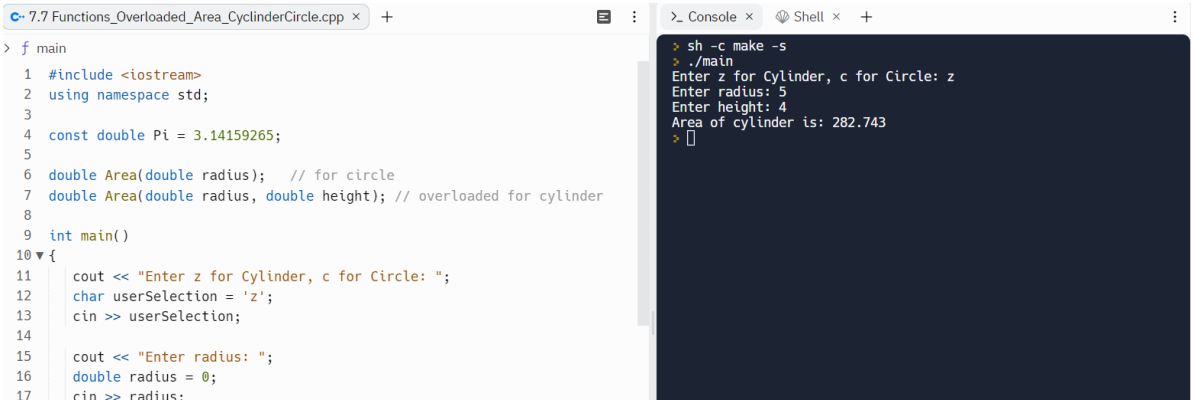
In this example, we can give a default value of our chosing to something otherwise another value is specified, for example, In the first example, the second parameters defaults to the value of 3.14, the function area() can still be invoked even if the second parameter didn't exist. Meanwhile in the second example when a new value of pi is given, then the second parameter will be used. This program demonstrates that we can program functions that contain default values for parameters that can be overriden by a user-supplied value. In both the example, the user both entered the same radius, but the area was different due to value of pi.

Example 7.7

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

const double Pi = 3.14159265;

double Area(double radius);   // for circle
double Area(double radius, double height); // overloaded for cylinder

int main()
{
   cout << "Enter z for Cylinder, c for Circle: ";
   char userSelection = 'z';
   cin >> userSelection;

   cout << "Enter radius: ";
   double radius = 0;
   cin >> radius;

   if (userSelection == 'z')
   {
      cout << "Enter height: ";
      double height = 0;
      cin >> height;

      // Invoke overloaded variant of Area for cylinder
      cout << "Area of cylinder is: " << Area (radius, height) << endl;
   }
   else
      cout << "Area of circle is: " << Area (radius) << endl;

   return 0;
}

// for circle
double Area(double radius)
{
   return Pi * radius * radius;
}

// overloaded for cylinder
```

```
double Area(double radius, double height)
{
   // reuse the area of circle
   return 2 * Area (radius) + 2 * Pi * radius * height;
}
```

Output



心得

This programs shows how we can use functions with the same name and return type but different parameters, which is known as overloaded functions. We can use this where a function with a particular name that has a particular output that might to be invoked by different sets of parameters, in this example it is a program that computers the area off a circle and area of a cylinder, the function that computes the area of the circle needs a radius as parameter, meanwhile for cylinder aside from the radius, the height is also needed. C++ enables programmer to define two overloaded functions with the same name and return type.