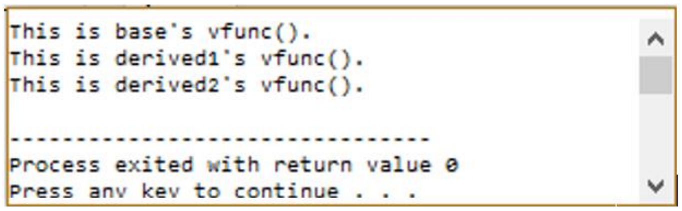
Notice: Return your C++ source codes into Oma before deadline. Only so you can get credits from this exercise. You can't return these source codes after deadline.

1. **Virtual functions** (http://www.learncpp.com/cpp-tutorial/122-virtual-functions/). Implement base class **base** which contains public virtual function **vfunc()**.This function prints text "This is base's vfunc().". Further implement derived class **derived1** which contains public function **vfunc()**. This function prints text " This is derived1's vfunc().".Further implement derived class **derived2** which contains public function **vfunc()**.This function prints text " This is derived2's vfunc().". Implement main function where you define variables

```
base *p, b;
derived1 d1;
derived2 d2;
```

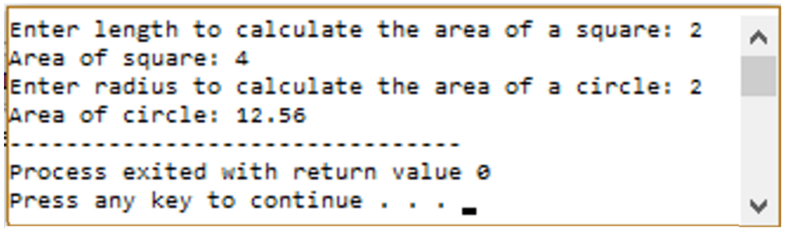
After that you have to make statements so that the output is like in figure 1.



```
This is base's vfunc().
This is derived1's vfunc().
This is derived2's vfunc().

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Figure 1. Sample print in Dev C++ -program

2. **Abstract Class and Pure Virtual Function**. Implement abstract class **Shape** with 1 protected property **l**. Implement one public method getData which requests a value to property **l**. Make also one pure virtual function **calculateArea**[1]. Further implement derived class **Square** which contains public function **calculateArea()**. It returns the area of square (l*l). Further implement derived class **Circle** which contains public function **calculateArea()**.It returns the area of circle (3.14*l*l). In main class define one circle and one square type variable. After that you have to make statements so that the output is like in figure 2.

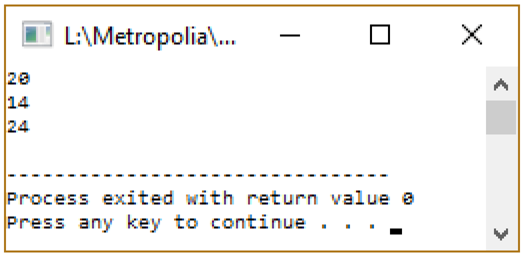


```
Enter length to calculate the area of a square: 2
Area of square: 4
Enter radius to calculate the area of a circle: 2
Area of circle: 12.56
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Figure 2. Sample print in Dev C++ -program

[1] A virtual function whose declaration ends with *=0* is called a pure virtual function.

3. **Pure Virtual Function**. Implement abstract class **number** with 1 protected property **val**. Implement one public method **setval** which requests a value to property **val**. Make also one pure virtual function **show**[2]. Further implement derived class **hextype** which contains public function **show()**.It returns the value of property **val** in hexadecimal form. Further implement derived class **dectype** which contains public function **show()**. It returns the value of property **val** in decimal form. Furthermore implement derived class **octtype** which contains public function **show()**. It returns the value of property **val** in octal form. In main class define one dectype variable, one octtype variable and one hextype variable. Make setval method call of each objects and put parameter value 20 of each call. Furthermore make **show** method call of each objects. Sample output is in figure 3.



```
20
14
24

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Figure 3. Sample print in Dev C++ -program

4. **Generic Functions**. The general form of a template function definition is

```
template <class Ttype> ret-type func-name(parameter list)
{
    // body of function
}
```

Implement a template which starts with clause

```
template <class X> void swapargs(X &a, X &b)
```

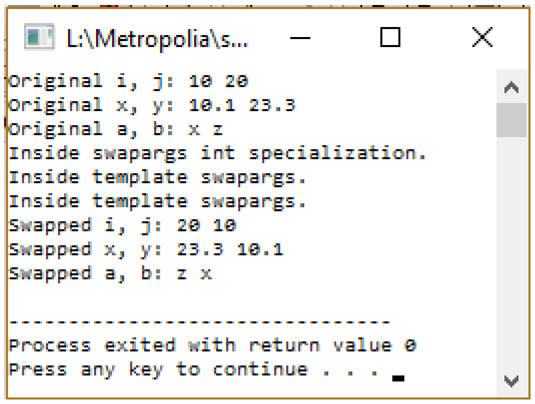
You have to implement the body of that function which swaps the values of variable a and variable b. Test your program with values

```
int i=10, j=20;
double x=10.1, y=23.3;
char a='x', b='z';
```

Sample print is in figure 4.

< continued >

[2] A virtual function whose declaration ends with *=0* is called a pure virtual function.



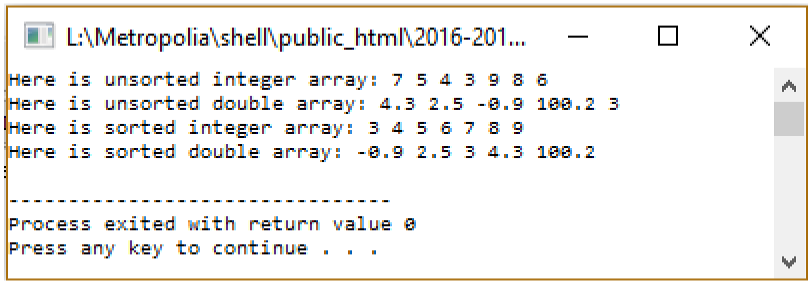
```
Original i, j: 10 20
Original x, y: 10.1 23.3
Original a, b: x z
Inside swapargs int specialization.
Inside template swapargs.
Inside template swapargs.
Swapped i, j: 20 10
Swapped x, y: 23.3 10.1
Swapped a, b: z x

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Figure 4. Sample print in Dev C++ -program

5. **Generic sort**. Implement generic function bubble which contains two parameters. First parameter is a pointer to array to be sorted and second parameter is number of items in array. Implement the body of this function. In main function you have to test the whole program with next two arrays.

```
int iarray[7] = {7, 5, 4, 3, 9, 8, 6};
double darray[5] = {4.3, 2.5, -0.9, 100.2, 3.0};
```

Sample print is in figure 5.



```
Here is unsorted integer array: 7 5 4 3 9 8 6
Here is unsorted double array: 4.3 2.5 -0.9 100.2 3
Here is sorted integer array: 3 4 5 6 7 8 9
Here is sorted double array: -0.9 2.5 3 4.3 100.2

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

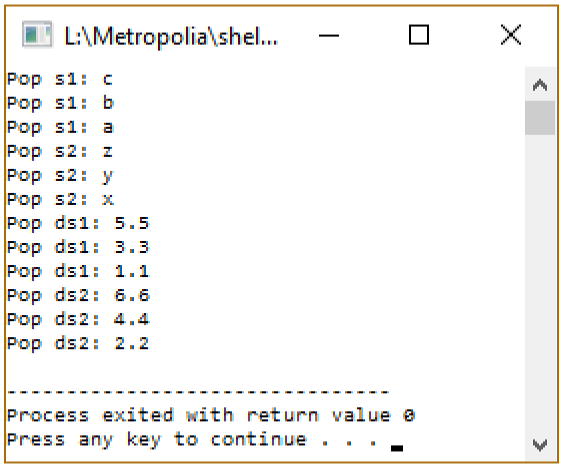
Figure 5. Sample print in Dev C++ -program

< continued >

6. **Generic stack**. Implement generic stack class **stack** which contains two parameters. First parameter holds the stack and second parameter is index of top of the stack. You can use next definitions

```
StackType stck[SIZE]; // holds the stack
int tos; // index of top-of-stack
```

You have to make public function **stack()** which initializes the stack. Further you have to make public function **push()** which puts the object on stack. Object is given in parameter. Further you have to make public function **pop()** which takes the top object from stack. In main function you have to create two character stacks. You have to put characters ‘a’, ‘b’ and ‘c’ on stack **s1** and characters ‘x’, ‘y’ and ‘z’ on stack **s2**. two test the whole program with next two arrays. Furthermore you have to create two double stacks. You have to put doubles 1.1, 3.3 and 5.5 on stack **ds1** and characters 2.2, 4.4 and 6.6 on stack **ds2**. Finally you have to print values of each stack in order **s1**, **s2**, **ds1** and **ds2**. Sample print is in figure 5.



```
Pop s1: c
Pop s1: b
Pop s1: a
Pop s2: z
Pop s2: y
Pop s2: x
Pop ds1: 5.5
Pop ds1: 3.3
Pop ds1: 1.1
Pop ds2: 6.6
Pop ds2: 4.4
Pop ds2: 2.2

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Figure 6. Sample print in Dev C++ -program