```
/* Thanks to former CS 31 TA Kung-Hua Chang for the set of practice
problems and solutions.
Ref: Practice Problems for C++ Beginners: Moving Beyond the Basics,
by Dr. Kung-Hua Chang. */

/*
Q1: What is the output of the program below given the inputs?
123456
105000
Software Engineer
*/

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

struct employee  // we can replace struct by class, but...
{
  int ID;
  double salary;
  string jobtitle;
};  // Remember to put semicolon here...

int main()
{
  employee emp;

  cout << "Please enter empoloyee information:\n";
  cout << "ID Number: ";
  cin >> emp.ID;

  cout << "Salary: ";
  cin >> emp.salary; cin.ignore(1000, '\n');

  cout << "Job title: ";
  getline(cin, emp.jobtitle);

  cout << "There is one employee:\n";

  cout << "ID Number: " << emp.ID << endl;
  cout << "Salary: " << emp.salary << endl;
  cout << "Job Title: " << emp.jobtitle << endl;
}
```

```
/* Q1-solution:
Output:
There is one employee:
ID Number: 123456
Salary: 105000
Job Title: Software Engineer
*/

/*
Q2: What is the output of the program below given the inputs?
123456
105000
Software Engineer
*/

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

struct employee  // we can replace struct by class, but...
{
  int ID;
  double salary;
  string jobtitle;
};  // Remember to put semicolon here...

int main()
{
  employee emp_a;
  employee *emp = &emp_a;

  // The arrow, ->, is used with a pointer to access the members of a
struct/class. emp->ID is equivalent to (*emp).ID

  cout << "ID Number: ";
  cin >> emp->ID;

  cout << "Salary: ";
  cin >> (*emp).salary;

  cin.ignore(1000, '\n');
  cout << "Job title: ";
  getline(cin, emp->jobtitle);

  cout << "There is one employee:\n";
```

```
  cout << "ID Number: " << emp->ID << endl;
  cout << "Salary: " << emp->salary << endl;
  cout << "Job Title: " << emp->jobtitle << endl;
}

/* Q2-solution:
Output:
There is one employee:
ID Number: 123456
Salary: 105000
Job Title: Software Engineer
*/

/*
Q3: What is the output of the program below given the inputs?
123456
105000
Software Engineer
*/

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

struct employee  // we can replace struct by class, but...
{
  int ID;
  double salary;
  string jobtitle;
};  // Remember to put semicolon here...

void getInput(employee emp) //
{
  cout << "ID Number: ";
  cin >> emp.ID;

  cout << "Salary: ";
  cin >> emp.salary;

  cin.ignore(1000, '\n');
  cout << "Job title: ";
  getline(cin, emp.jobtitle);
}

int main()
```

```
{
  employee emp = {0, 0, ""};  // initialization
  // same as: emp.ID = 0; emp.salary = 0; emp.jobtitle = "";

  getInput(emp);

  cout << "There is one employee:\n";
  cout << "ID Number: " << emp.ID << endl;
  cout << "Salary: " << emp.salary << endl;
  cout << "Job Title: " << emp.jobtitle << endl;
}

/* Q3-solution:
Output:
There is one employee:
ID Number: 0
Salary: 0
Job Title:

***NOTE***
The getInput() function should take as input the argument "employee
&emp" to pass by reference, so that the values obtained from the user
inputs can be stored to the correct struct variable.
*/


/*
Q4: What is the output of the program below given the inputs?
123456
105000
Software Engineer
246802
130000
Senior Software Engineer
*/

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

struct employee  // we can replace struct by class, but...
{
  int ID;
  double salary;
  string jobtitle;
```

```
};  // Remember to put semicolon here...

int main()
{
  employee emp[2];
  // employee *emp = new employee[2];

  for (int i = 0; i < 2; i++)
  {
    cout << "ID Number: ";   cin >> emp[i].ID;
    cout << "Salary: ";    cin >> emp[i].salary;    cin.ignore(1000,
'\n');
    cout << "Job title: ";    getline(cin, emp[i].jobtitle);
  }

  cout << "There are 2 employees:\n";

  for (int i = 0; i < 2; i++)
  {
    cout << "Employee #" << i + 1 << ":" << endl;
    cout << "ID Number: " << emp[i].ID << endl;
    cout << "Salary: " << emp[i].salary << endl;
    cout << "Job Title: " << emp[i].jobtitle << endl;
  }
}

/* Q4-solution:
Output:
There are 2 employees:
Employee #1:
ID Number: 123456
Salary: 105000
Job Title: Software Engineer
Employee #2:
ID Number: 246802
Salary: 130000
Job Title: Senior Software Engineer
*/


/*
Q5: What is the output of the program below given the inputs?
123456
105000
Software Engineer
```

```
246802
130000
Senior Software Engineer
*/

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

struct employee  // we can replace struct by class, but...
{
  int ID;
  double salary;
  string jobtitle;
};  // Remember to put semicolon here...

// same as employee *emp or employee emp[]
void getInput(employee emp[2])
{
  for (int i = 0; i < 2; i++)
  {
    cout << "ID Number: ";   cin >> emp[i].ID;
    cout << "Salary: ";    cin >> emp[i].salary;    cin.ignore(1000,
'\n');
    cout << "Job title: ";    getline(cin, emp[i].jobtitle);
  }
}


int main()
{
  employee emp[2];

  getInput(emp);

  cout << "There are 2 employees:\n";

  for (int i = 0; i < 2; i++)
  {
    cout << "Employee #" << i + 1 << ":" << endl;
    cout << "ID Number: " << emp[i].ID << endl;
    cout << "Salary: " << emp[i].salary << endl;
    cout << "Job Title: " << emp[i].jobtitle << endl;
  }
}
```

```
/* Q5-solution:
Output:
There are 2 employees:
Employee #1:
ID Number: 123456
Salary: 105000
Job Title: Software Engineer
Employee #2:
ID Number: 246802
Salary: 130000
Job Title: Senior Software Engineer
*/

/*
Q6: What is the output of the program below given the inputs?
*/

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

struct employee  // we can replace struct by class, but...
{
public:
  void outputID() { cout << ID << endl;}
  int ID;
  double salary;
};  // Remember to put semicolon here...

int main()
{
  employee emp[5];

  // Before we rotate left at the first position

  for (int i = 0; i < 5; i++)
  {
    emp[i].ID = i;
    emp[i].salary = 10000 * i + 50000;
    emp[i].outputID();
  }

  cout << "=====" << endl;
```

```
  // Now we want to rotate left at the first position

  employee temp = emp[0];
  for (int i = 0; i < 4; i++)
    emp[i] = emp[i+1];
  emp[4] = temp;

  for (int i = 0; i < 5; i++)
    emp[i].outputID();
}

/* Q6-solution:
Output:
0
1
2
3
4
=====
1
2
3
4
0
*/


/*
Q7: What is the output of the program below given the inputs?
123456
105000
Software Engineer
*/

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

class employee
{
public:
  void input();      // public member function
  void output();     // public member function
private:
  int ID;            // private data member
```

```
  double salary;     // private data member
  string jobtitle;   // private data member
};

void employee::input()
{
  cout << "ID Number: ";   cin >> ID;
  cout << "Salary: ";    cin >> salary;    cin.ignore(1000, '\n');
  cout << "Job title: ";    getline(cin, jobtitle);
}

void employee::output()
{
  cout << "There is one employee:\n";
  cout << "ID Number: " << ID << endl;
  cout << "Salary: " << salary << endl;
  cout << "Job Title: " << jobtitle << endl;
}

int main()
{
  employee emp;

  emp.input();
  emp.output();
}

/* Q7-solution:
Output:
There is one employee:
ID Number: 123456
Salary: 105000
Job Title: Software Engineer
*/


/*
Q8: If the following program doesn't compile, why not? If it does
compile, what is the output of the program below given the inputs?
123456
105000
Software Engineer
*/

#include <iostream>
```

```cpp
#include <string>
using namespace std;

class employee
{
  int ID;
  double salary;
  string jobtitle;
};

int main()
{
  employee *emp = new employee;
  // This also works: employee *emp = new employee();

  cout << "ID Number: ";
  cin >> emp->ID;
  cout << "Salary: ";
  cin >> (*emp).salary;
  cin.ignore(1000, '\n');
  cout << "Job title: ";
  getline(cin, emp->jobtitle);

  cout << "There is one employee:\n";

  cout << "ID Number: " << emp->ID << endl;
  cout << "Salary: " << emp->salary << endl;
  cout << "Job Title: " << emp->jobtitle << endl;
}

/* Q8-solution:
Compilation error because ID, salary, jobtitle data memebers are not
accessible from outside employee object as they are private data
members. To resolve this isue, use the publi keyword like:
class employee
{
public:
  int ID;
  double salary;
  string jobtitle;
}
*/


/*
```

```
Q9: What is the output of the program below?
*/

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

struct employee
{
  int ID;
  double salary;
  string jobtitle;
};

int main()
{
  employee emp1[2];
  // An employee struct array with 2 cells.
  // Each cell is an employee object.

  employee *emp2 = new employee[2];
  // An employee* pointer points to the starting address of the
  // employee struct array with 2 cells.

  employee *emp3[2];
  // emp3[0] stores an employee struct pointer (points to a random
place)
  // emp3[1] also points to a random place.

  cout << "Static array: size = " << sizeof(emp1[0]) << endl;

  cout << "Dynamic array: size = " << sizeof(emp2[0]) << endl;

  cout << "Array sotring only pointers: size = " << sizeof(emp3[0]) <<
endl;
}

/* Q9-solution:
NOTE: you should see different values on different machines.
The idea is to show the difference in sizes between the array of
pointers ad the array of objects.

In g++ on MacOS 64-bit machine:
Output:
Static array: size = 40
```

```
Dynamic array: size = 40
Array sotring only pointers: size = 8
*/


/* Q10: If the following program doesn't compile, why not? If it does
compile, what is the output when it is run?
*/

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

class employee
{
public:
  employee()
  {
    ID = 0;
    salary = 0;
    jobtitle = "";
  }
  employee(int ID, double salary, string jobtitle)
  {
    this -> ID = ID;
    this -> salary = salary;
    this -> jobtitle = jobtitle;
  }
  ~employee()
  {
    cout << "This is destructor!!" << endl;
  }
  void Output();
private:
  int ID;
  double salary;
  string jobtitle;
};
void employee::Output()
{
  cout << "ID Number: " << ID << endl;
  cout << "Salary: " << salary << endl;
  cout << "Job Title: " << jobtitle << endl;
}
```

```
int main()
{
  employee A;                                                  // (1)
  employee B();                                                // (2)
  employee C = {0, 0, ""};                                     // (3)
  employee D(1234, 90000.0, "Software Engineer");              // (4)
  employee *E = new employee;                                  // (5)
  employee *F = new employee();                                // (6)
  employee *G = new employee(5432, 95000.0, "Manager");        // (7)
  A.Output();                                                  // (8)
  B.Output();                                                  // (9)
  C.Output();                                                  // (10)
  D.Output();                                                  // (11)
  E->Output();                                                 // (12)
  F->Output();                                                 // (13)
  G->Output();                                                 // (14)
}

/* Q10-solution:
(9) cause a compilation error because (2) is to declare the prototype of
a function B taking 0 arguments and returning an employee object. In
other words, the compiler thinks that the function implementation for
the function B is after the main() fuction like:
int main()
{
  employee B();  // the prototype of the function B
}
employee B()     // the implementation of the function B
{
  employee b;
  return b;
}
Thus, B is not an object, so it cannot use B.Output().

(3) cause a compilation error because ID, slary, jobtitle are defined as
private members.
*/


/* Q11: What is the output of the program below?
The general way to code a class is to
(1) Hide information: leave data memeber in private section
(2) Declare accessor and mutator functions to access private members.
      - Accessor: only read the private data members out.
      - Mutator: change the private data members.
```

```
*/

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

class employee
{
public:
  // Is is okay NOT to declare your own constructor and destructor?

  void setValue(int theID, double theSalary)
  {    // This is a mutator function
    ID = theID; salary = theSalary;
  }

  //These two member function below are accessor functions
  int getID()
  {
    return ID;
  }
  double getSalary();

private:
  int ID;
  double salary;
};

double employee::getSalary()
{
  return salary;
}

int main()
{
  employee emp[2];

  emp[0].setValue(1, 100000.0);
  emp[1].setValue(2, 110000.0);

  cout << "ID1: "       << emp[0].getID()
       << " Salary1 = " << emp[0].getSalary() << endl;

  cout << "ID2: "       << emp[1].getID()
       << " Salary2 = " << emp[1].getSalary() << endl;
```

```
}

/* Q11-solution:
Output:
ID1: 1 Salary 1 = 100000
ID1: 2 Salary 2 = 110000

***NOTE***: if you do not declare any constructor/destructor, then the
C++ compiler will take default constructor(s) / destructor for you. If
you declare a constructor/destructor, then C++ compiler will not make a
default one for you.
*/


/*
Q12: What is the output of the program below?
*/

#include <iostream>
#include <string>
using namespace std;
class employee
{
public:
  employee()
  {
    ID = 0;
    salary = 0;
    jobtitle = "";
  }

  void setData(int ID, double salary, string jobtitle)
  {
    this -> ID = ID;
    this -> salary = salary;
    this -> jobtitle = jobtitle;
  }

  int getID() const { return ID; }

  ~employee()
  {
    cout << ID << " got laid off!!" << endl;
  }
```

```
  void Output()
  {
    cout << "ID number: " << ID << endl;
    cout << "Salary: " << salary << endl;
    cout << "Job Title: " << jobtitle << endl;
  }

private:
  int ID;
  double salary;
  string jobtitle;
};

class company
{
public:
  company() {}
  company(string name, int numEmp)
  {
    this -> name = name;
    emp = new employee[numEmp];
    for (int i = 0; i < numEmp; i++)
    {
      emp[i].setData(i+1, 80000, "Software Engineer");
      cout << emp[i].getID()
           << " got hired to join the company" << endl;
    }
  }
  ~company()
  {
    cout << this -> name << " company goes bankrupt!!" << endl;
    delete [] emp;
  }
private:
  employee *emp;
  string name;
};

int main()
{
  company comp("ABC", 2);
}

/* Q12-solution:
Output:
```

```
1 got hired to join the company
2 got hired to join the company
ABC company goes bankrupt!!
2 got laid off!!
1 got laid off!!
*/
```