Employee.h:

```
#include <iostream>
#include <string>
#include "employee.h"
using namespace std;

#ifndef WAITER
#define WAITER
class waiter : public employee
{
public:
	waiter(long employeeID = 0, string last = "", string first = "",
		char employeeType = ' ', double salary = 3000, double profit = 0, double
tip = 0, string experience = "");
	virtual void display(ostream& out) const;
	virtual double Calculate_salary(double profit, double tip);
private:
	string experience;
	double sal = getSalary(); //accessor
};
inline waiter::waiter
(long employeeID, string last, string first, char employeeType,
	double salary,  double profit, double tip, string experience)
	:employee(employeeID, last, first, employeeType, salary, profit, tip),
experience(experience)
{}
inline void waiter::display(ostream& out)const
{
	out << "employID " << "last name " << "first name " << "status " << "salary " <<
"profit " << "tip " <<"experience" << endl;
	employee::display(out);
	out << experience << endl;
}
inline double waiter::Calculate_salary(double profit, double tip) // waiter only have tip
as additional salary
{
	sal = sal + tip;
	return sal;
};

#endif
```

Owner.h :

```
#include <iostream>
#include "employee.h"
#include <string>
using namespace std;

#ifndef OWNER
#define OWNER
class owner : public employee
{
public:
	owner(long employeeID = 0, string last = "", string first = "",
		char employeeType = ' ', double salary = 15000, double profit = 0, double
tip = 0);
	virtual void display(ostream& out) const;
	virtual double Calculate_salary(double profit, double tip);
private:
	double sal = getSalary(); // accessor
};
inline owner::owner
(long employeeID, string last, string first, char employeeType,
	double salary, double profit, double tip)
	:employee(employeeID, last, first, employeeType, salary, profit, tip)
{}
inline void owner::display(ostream& out)const
{
	out << "employID " << "last name " << "first name " << "status " << "salary " <<
"profit " << "tip" << endl;
	employee::display(out);
	out << endl;
}
inline double owner::Calculate_salary(double profit, double tip)  // owner has 60 % of
the profit as the additional slary
{
	if (profit >= -25000)
		sal = sal + 0.6 * profit;
	else
		sal = 0;
	return sal;
};

#endif
```

Chef.h

```
#include <iostream>
#include <string>
#include "employee.h"
using namespace std;

#ifndef CHEF
#define CHEF
class chef : public employee
{
public:
	chef(long employeeID =0, string last = "", string first = "",
		char employeeType = ' ', double salary = 10000, string cuisine = "", double
profit = 0, double tip = 0);
	virtual void display(ostream& out) const;
	virtual double Calculate_salary(double profit, double tip);
private:
	string cuis;
	double sal = getSalary(); //accessor
};
inline chef::chef
(long employeeID, string last, string first, char employeeType,
	double salary, string cuisine, double profit, double tip)
	:employee(employeeID, last, first, employeeType, salary, profit, tip),
cuis(cuisine)
{}
inline void chef::display(ostream& out)const //
{
	out << "employID " << "last name " << "first name " << "status " << "salary " <<
"profit " << "tip  " << "cuisine" << endl;
	employee::display(out);
	out << cuis << endl;
}
inline double chef::Calculate_salary(double profit, double tip) // chef has 20%
additional salary from the profit
{
	if (profit >= -25000)
		sal = sal + 0.2 * profit;
	else
		sal = 5000;
	return sal;
};

#endif
```

Waiter.h:

```
#include <iostream>
#include <string>
#include "employee.h"
using namespace std;

#ifndef WAITER
#define WAITER
class waiter : public employee
{
public:
	waiter(long employeeID = 0, string last = "", string first = "",
		char employeeType = ' ', double salary = 3000, double profit = 0, double
tip = 0, string experience = "");
	virtual void display(ostream& out) const;
	virtual double Calculate_salary(double profit, double tip);
private:
	string experience;
	double sal = getSalary(); //accessor
};
inline waiter::waiter
(long employeeID, string last, string first, char employeeType,
	double salary,  double profit, double tip, string experience)
	:employee(employeeID, last, first, employeeType, salary, profit, tip),
experience(experience)
{}
inline void waiter::display(ostream& out)const
{
	out << "employID " << "last name " << "first name " << "status " << "salary " <<
"profit " << "tip " <<"experience" << endl;
	employee::display(out);
	out << experience << endl;
}
inline double waiter::Calculate_salary(double profit, double tip) // waiter only have tip
as additional salary
{
	sal = sal + tip;
	return sal;
};

#endif
```

Driver.cpp

```cpp
#include <iostream>
#include "employee.h"
#include "chef.h"
#include "owner.h"
#include "waiter.h"
#include <list>
using namespace std;
int main()
{
	double profit1, tip1;
	cout << "please enter the profit and tip:" << endl;
	cin >> profit1;
	cin >> tip1;
	employee* p;
	list<employee*> empList;
	p = new owner(11111, "tsire", "mariami",'o', 15000, profit1, tip1);
	empList.push_back(p);
	p = new chef(22222, "Gordon", "Ramsey", 'c', 10000, "french", profit1, tip1);
	empList.push_back(p);
	p = new chef(22222, "Oliver", "Jamie", 'c', 10000, "Italian", profit1, tip1);
	empList.push_back(p);
	p = new waiter(33333, "Oliver", "Anne", 'w', 3000, profit1, tip1, "1w");
	empList.push_back(p);
	p = new waiter(33333, "Puck", "Marrie", 'w', 3000, profit1, tip1 , "3w");
	empList.push_back(p);
	p = new waiter(33333, "Rayolds", "Nina", 'w', 3000, profit1, tip1, "2w");
	empList.push_back(p);

	for (list<employee*>::iterator it = empList.begin();
		it != empList.end(); it++)
	{
		p = *it;
		cout << *p << endl;
		cout << "total salary" << endl;
		cout << p->Calculate_salary(profit1, tip1) << endl << endl;
	}


}
```