Lab 5

Ex 5.7.2.1

1 . Employee Stack

You are the manager of data processing department that has the task of hiring and laying off employees. Seniority is important because you will lay off employees in the reverse order that they were hired (a Last In. First Out structure). This ensures that employees who were hired first will be the last to be laid off. Using the Stack class from the Chapter 2 exercises as a model, create a class that represents a stack of employees.

An employee object should contain the following data members: ID, last name, first name, job title, years with company. Write a test program with a menu offering the following choices:

1. Hire new employee .
2. Let the user input a complete employee record.
3. Lay off employee (s).
4. Let the user select the number of employees to be laid off. (The program should display all information about the employees that are removed from the stack.)
5. List all employees.
6. Exit program.

1-The employee header file:

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include<string>
using namespace std;

class Employee
{
   public:
      Employee();
      Employee(long i, string fname,string lname,string job, int y);
      void setFirstName(string &fname);
      void setLastName(string &lname);
      string & getFirstName();
      string & getLastName();
       void input();
       friend istream &operator>>(istream & in , Employee &E);
       friend ostream &operator<<(ostream & out , Employee &E);

   private:
      long id;
      string first_name;
      string last_name;
      string job_title;
```

```
    int years;

};

#endif // EMPLOYEE_H
```

2- The employee source file:

```
#include "Employee.h"
#include<iostream>
#include<string>
using namespace std;
Employee::Employee()
{
   id = 0;
   last_name="";
   first_name="";
   job_title="";
   years = 0;
}
 Employee::Employee(long i, string fname,string lname,string job, int y)
 {
    id=i;
    first_name=fname;
    last_name=lname;
    job_title=job;
    years=y;

 }
 void Employee::setFirstName(string &fname)
 {
    first_name=fname;
 }

void Employee::setLastName(string &lname)
{
   last_name=lname;
}

string & Employee:: getFirstName()
{
   return first_name;
}

string &  Employee::getLastName()
{
   return last_name;

}
void Employee:: input()
{
   cout << "Input an Employee record:\n ID number  " ;
```

```
   cin >> id;
   cin.ignore( ) ;
   cout << "First name  " ;
   getline(cin,first_name);
   cout << "Last name  " ;
   getline( cin,last_name);
   cout << "Job title  ";
  getline(cin,job_title);
  cout<<"years";
  cin>>years;

}

istream &operator>>(istream & in , Employee &E)
{
   in>>E.id>>E.last_name>>E.job_title>>E.years;
   return in;
}

ostream &operator<<(ostream & out , Employee &E)
{
   out<<E.id<<"\t"<<E.first_name<<"\t"<<E.last_name<<"\t"<<E.job_title<<"\t"<<E.years<<endl;
   return out;
}
```

3- The stack header file:

```
#ifndef STACK_H
#define STACK_H
#include"Employee.h"
const int SIZE=30;
class Stack
{
   public:
      Stack();

      void HireNewEmployee();
      Employee  LayOffEmployee();
      void LayOffMultipleEmployees();
      void ListAllEmployees();

   private:
      int top;
      Employee emp[SIZE];

      void Push(  Employee & emp);
      Employee& Pop();
      bool Empty();
      bool Full();
};

#endif // STACK_H
```

3- The stack source file:

```
#include "Stack.h"
#include<iostream>
#include<iomanip>
using namespace std;
Stack::Stack()
{
   top=0;
}

bool Stack::Empty()
{
   return top<=0;
}

bool Stack::Full()
{
   return top>=SIZE;
}

void Stack::Push(Employee & E)
{
   emp[top++]=E;
}

Employee& Stack::Pop()
{
   return emp[--top];
}


 void Stack::HireNewEmployee()
 {
    if(!Full())
    {
        cout << "Hiring new employee.\n" << endl;
        Employee E;
        E.input();
        Push( E);
        cout << "The employee has been hired.\n" << endl;
        }
   else
       cout << "Can't hire more employee  no available space try latter"<<endl;
}


Employee Stack::LayOffEmployee()
{


   if( !Empty())
      return Pop();
  else
```

```
        cout << "No employess remain,  none one is laid off."<<endl;
        Employee e;
        return e;


}

void Stack::LayOffMultipleEmployees()
{
   int count = top;
   int layoffCount;

   cout << " How many employees do you want to lay off";
   cin >> layoffCount;

   if( layoffCount > count)
   cout << "There are only  " << count<<" employees available.can't complete.\n" << endl;

   else
      for(int i = 0; i < layoffCount; i++)
      {
         Employee E = Pop();
         cout << "Laying off employee:  " << E << endl;
      }
}


void Stack::ListAllEmployees()
{
   if(!Empty())
   {
      int to=top;
      while(to!=0)
      {

            Employee E = emp[--to];
            cout<<E<<endl;
      }
   }
   else
      cout<<"no employee to list"<<endl;

}
```

5- main source file

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

```
unsigned menu()
{

   int choice;
   cout << "\n-----------------------------------------------\n "
        <<"press\n1  Hire new employee\n2  Input an employee record\n"
        <<"3  Lay off one employee\n4  Select number of employees to lay off\n"
        <<"5  List all employees\n6  Exit program\n\n ";

   cin >> choice;
   return choice;
}
int main()
{
   Stack S;
   Employee emp;

   while( true)
      {
         int choice =menu();
         cout<<endl;
         switch( choice)
         {
            case 1:
               S.HireNewEmployee();
               break;
             case 2:
                emp.input();
                break;
            case 3:
               emp=S.LayOffEmployee();
               cout << "Laid off:"<<emp<<endl;
               break;
             case 4:
               S.LayOffMultipleEmployees();
               break;
            case 5:
               S.ListAllEmployees();
               cout << endl;
               break;
            case 6:
               return 0;
         }
      }

   return 0;
}
```