

CIS657 Spring 2019

Lab Assignment 2

Submitted by –

Name – Anish Nesarkar

SUID – 368122582

Subject – Operating Systems

Lab 2 Assignment

CIS657 Spring 2019

Assignment Disclosure Form

Assignment #: 2

Name: Anish Nesarkar

1. Did you consult with anyone other than instructor or TA/grader on parts of this assignment?

If Yes, please give the details.

2. Did you consult an outside source such as an Internet forum or a book on parts of this assignment?

If Yes, please give the details.

I assert that, to the best of my knowledge, the information on this sheet is true.

Signature: _____ Anish Nesarkar _____

Date : 2/15/2019

Design:

An Employee Management Software is implemented for a company XYZ. The details of the employee working in the company are stored in employee.dat file. The employee.dat file consists of the following employee records:

- Name
- Unique ID
- Position
- Department
- Hourly pay rate

The above data is read from employee.bat file if it exists. Different methods are defined that perform particular functionalities like adding new record to existing employee records, searching for an employee in the list, updating existing employee information, provision to delete an employee record, scheduling weekly jobs to the existing employees. The user is allowed to perform the above actions. When the user exits the program, all the actions performed on the employee records are stored back to the employee.bat file.

Requirement 1: File Opening and Reading employee.dat

- The employee.bat file is stored in threads folder of Nachos.
- A function is defined in threadtest.cc to read from the file if it exists.
- An employee object pointer is created to store data from every line of the file until end of the file.
- The characters from the line are appended to a string until a space is detected.
- The corresponding string is converted to integer and stored in employee object using setter methods.
- After storing all the information in the employee object, the employee object is appended to an employee list.

Requirement 2: Enter New Record

- In this requirement, a new employee record should be added to the employee list.
- An employee object pointer is created.
- The ID of the employee is set to value one more than the existing employee ID and set to the employee object using setter method.
- The employee Name is set using setter method in employee object.
- The position is set using setter method in employee object.
- The department is set using setter method in employee object.
- The hourly rate is set using setter method in employee object.
- The employee object is the appended to the employee list.

Requirement 3: Display all employees

- In this requirement, all employees record is displayed using the Apply function from list.cc
- A display function is passed as an argument to the Apply function.
- The apply function iterates through the list and prints each employee records.

Requirement 4: Search Employee(s)

- In this requirement, the user is asked to search for an employee(s) using employees user ID or employee department.
- After selection, if the user wants to search employee using ID, the user enters the employees ID and the ID is checked with the employee objects IDs iterating through the list. The employee record is displayed using Apply function.
- If the user selects search by department, all the employees in that department are printed by iterating through the list using Apply function from list.cc.

Requirement 5: Update Employee Information

- The existing employee details are updated in this function.
- The user is asked for employee ID to be updated.
- The employee object is found using list iterator whose ID is equal to the ID entered by the user.
- Different options are displayed to choose what to update like Name, Department, position and hourly rate.
- The new information is over written on the existing record.

Requirement 6: Delete Employee Information

- This function is used to delete the existing employee record.
- The user is asked for the employee ID to be deleted.
- The user is again asked for confirmation whether to delete the employee info.
- A list iterator is created to locate the employee object whose ID is equal to the ID entered by the user.
- After finding the employee object, it is deleted from the memory and removed from the employee list.

Requirement 7: Scheduling Weekly jobs

- The list is sorted in the descending order of the Individual employee paycheck using the defined comparePay function.
- Randomly generated number of hours worked by employee in Employee class constructor is multiplied with the payrate of individual employee

Requirement 8: Saving to the File and Exit

- When the chooses to exit, a saveToFile() function is invoked to save the employee list to the file.
- After opening the file, an employee object is created.
- A list Iterator is used to iterate through the list and store each employee object to file until the list reaches its end.
- After storing the data to the employee.dat file, the file is closed.

Implementation:

1. Employee.cc

```
#include "string.h"
#include "kernel.h"
#include "main.h"
#include "thread.h"
#include "list.h"
#include "employee.h"
#include<iostream>

//Employee constructor that calculates number of hours worked by an employee
Employee::Employee() {
hoursWorked = (rand() % 21 + 20);
}

//return the ID of the employee
int Employee::getId()
{
    return id;
}

//set the ID of the employee
void Employee::setId(int Id)
{
    id = Id;
}

//return the name of the employee
string Employee::getName()
{

```

```
        return Name;
    }

    //set the name of the employee
    void Employee::setName(string n)
    {
        Name = n;
    }

    //return the department of the employee
    int Employee::getDept()
    {
        return Department;
    }

    //set the department of the employee
    void Employee::setDept(int dpt)
    {
        Department = dpt;
    }

    //return the position of the employee
    int Employee::getpos()
    {
        return position;
    }

    //set the position of the employee
    void Employee::setpos(int pos)
    {
        position = pos;
    }

    //return the hourly rate of the employee
    int Employee::gethrate()
    {
        return hourlyRate;
    }

    //set the hourly rate of the employee
    void Employee::sethrate(int hr)
    {
        hourlyRate = hr;
    }
}
```

```

//return the number of hours worked by the employee
int Employee::getHoursWorked()
{
    return hoursWorked;
}

//return the paycheck of the individual employee
int Employee::getpayCheck()
{
    return (hourlyRate * hoursWorked);
}

```

2. Employee.h

```

#include "string.h"
class Employee {
private:
    int id; // ID of the employee
    string Name; //Name of the employee
    int Department; //Department of the employee
    int position; //position of the employee
    int hourlyRate; //hourly rate of the employee
    int hoursWorked; // number of hours worked by the employee
public:
    Employee(); //Employee Constructor
    ~Employee(){} //Employee Destructor

    int getId(); //Return Employee ID
    void setId(int Id); //Set employee ID

    string getName(); //return employee name
    void setName(string n); //set employee name

    int getDept(); //return employee department
    void setDept(int dpt); //set employee department

    int getpos(); //return employee position
    void setpos(int pos); //set employee position

    int getthrate(); //return employee hourly rate
    void setthrate(int hr); // set employee hourly rate

    int getpayCheck(); // return individual employee pay check

```

```
int getHoursWorked(); // return number of hours worked by the employee
};
```

3. Threadtest.cc

a. Reading from the file

```
while (inFile.getline(str, 255, '\n')) {
    //creating pointer to the object
    Employee *e = new Employee();
    i = 0;
    //count gives the substrings positions
    count = 0;
    //run this loop until last character is null
    while (!(str[i-1] == '\0'))
    {
        //get the Substrings and concatenate to the string
        while (!((str[i] == '\0') || (isspace(str[i]))))
        {
            if (count == 0)
                text = text + str[i];
            else if(count == 1)
                text = text + str[i];
            else if (count == 2)
                text = text + str[i];
            else if (count == 3)
                text = text + str[i];
            else if (count == 4)
            {
                text = text + str[i];
            }
            i++;
        }
        //Set the Employee ID
        if (count == 0)
        {
            test = text.c_str();
            int tempid = atoi(test);
            e->setId(tempid);
            if(idMax <= tempid)
                idMax = tempid + 1;
        }
    }
}
```



```

        //Set the name
        else if (count == 1)
        {
            e->setName(text);
        }
        //Set the department
        else if (count == 2)
        {
            test = text.c_str();
            e->setDept(atoi(test));
        }
        //Set the position
        else if (count == 3)
        {
            test = text.c_str();
            e->setpos(atoi(test));
        }
        //Set the hourly rate
        else if (count == 4)
        {
            test = text.c_str();
            e->sethrate(atoi(test));
        }
        i++;
        count++;
        text = "";
    }
    //Append the employee object pointer to the list
    eList->Append(e);
}
}

```

b. Enter new Employee record

```

//create employee object pointer
Employee *e = new Employee();
e->setId(idMax);
idMax++;
//Set the employee name
cout << "Enter Employee Name" << endl;
string name;
cin >> name;

```

```

e->setName(name);
//Set the employee position
cout << "Enter Position" << endl;
int position;
cin >> position;
e->setpos(position);
//set the employee department
cout << "Enter Department" << endl;
int dept;
cin >> dept;
e->setDept(dept);
//set the employee hourly rate
cout << "Enter hourly rate" << endl;
int hr;
cin >> hr;
e->sethrate(hr);
//Append the employee object pointer to the list
eList->Append(e);

```

c. Display Employees using Apply function from the list

```

cout << endl;
cout << ">-----< Employee Record >-----<" << endl << endl;
cout << "Employee ID: "<< e->getId() << endl;
cout << "Employee Name: "<< e->getName() << endl;
cout << "Employee Department: "<< e->getDept() << endl;
cout << "Employee Position: "<< e->getpos() << endl;
cout << "Employee hourly rate: "<< e->gethrate() << endl <<endl << endl;

```

d. Search Employee by ID

```

Employee *e = new Employee();
//create List Iterator to iterate through the employee list
ListIterator<Employee*> *Ilist = new ListIterator<Employee*>(eList);
List<Employee*> *eListID = new SortedList<Employee*>(compare);
//Iterate through the list
while(!Ilist->IsDone())
{
    //get the current item in the list
    e = Ilist->Item();
    //check if entered ID is equal to the employee ID from the list
}

```

```

        if(id == e->getId())
        {
            //append the employee object pointer to the list
            eListID->Append(e);
            //display the employee record using Apply function
            eListID->Apply(displayEmployees);
            return;
        }
        //point to next object in the list
        Ilist->Next();
    }

    cout << "Employee entry does not exist" << endl << endl;

```

e. Displaying department employees

```

void printDeptEmployees(int dept)
{
    //create employee object pointer
    Employee *e = new Employee();
    //create list iterator to iterate through the list
    ListIterator<Employee*> *Ilist = new ListIterator<Employee*>(eList);
    //create list for storing employees in the department
    List<Employee*> *eListDept = new SortedList<Employee*>(compare);
    //iterate through the list
    while(!Ilist->IsDone())
    {
        //assign employee to current item in the list
        e = Ilist->Item();
        //Check if the entered department is present in the list
        if(dept == e->getDept())
        {
            //append the employee object pointer to the list
            eListDept->Append(e);
        }
        //point to next item in the list
        Ilist->Next();
    }
    //print the employees in the department using apply function
    eListDept->Apply(displayEmployees);
}

```

f. Search employees by ID and Department

```
//switch statement to search employee by ID or department
switch(select)
{
    case 1:
    {
        int id;
        cout << "Enter Employee ID" << endl;
        cin >> id;
        //function to search employee by ID
        SearchEmployeebyID(id);
        break;
    }
    case 2:
    {
        int dept;
        cout << "Enter Department number" << endl;
        cin >> dept;
        //function to search employee by department
        printDeptEmployees(dept);
        break;
    }
    default:
        cout << "Invalid Entry" << endl << endl;
}
}
```

g. Update Employee details

```
//iterate through the list to get the employee ID
while(!Ilist->IsDone())
{
    e = Ilist->Item();
    if(id == e->getId())
        break;
    Ilist->Next();
}
//Check if ID exists
if(id == 0)
{
    cout << "Employee Entry does not exist" << endl;
    return;
}
```

```

    }
    int select;
    //update the employee details
    while(select != 5)
    {
        cout << "1. Update Name\n2. Update Department\n3. Update Position\n4.
Update hourly rate\n5. Exit\n Select Option" << endl;
        cin >> select;
        cout << endl;
        switch(select){
            case 1: {
                cout << "Enter New Name" << endl;
                string name;
                cin >> name;
                //set employee new name
                e->setName(name);
                break;
            }
            case 2:{
                cout << "Enter New Department" << endl;
                int dept;
                cin >> dept;
                //set employee new department
                e->setDept(dept);
                break;
            }
            case 3:{
                cout << "Enter New position" << endl;
                int pos;
                cin >> pos;
                //set employee new position
                e->setpos(pos);
                break;
            }
            case 4:{
                cout << "Enter New hourly rate" << endl;
                int hrate;
                cin >> hrate;
                //set employee new hour rate
                e->sethrate(hrate);
                break;
            }
            case 5:{
                break;
            }
        }
    }
}

```

```

        default:
            cout << "Invalid Selection" << endl;
    }

}

//create employee temporary list
List<Employee*> *eListtemp = new SortedList<Employee*>(compare);
//create employee list iterator to iterate through the list
ListIterator<Employee*> *Ilisttemp = new ListIterator<Employee*>(eList);
//iterate through the list to sort the list after updating
while(!Ilisttemp->IsDone())
{
    //get the current item from the list
    e = Ilisttemp->Item();
    //append the employee object pointer to the list
    eListtemp->Append(e);
    //get the next item from the list
    Ilisttemp->Next();
}
eList = eListtemp;
}

```

h. Delete Employee

```

//Delete the employee object if "Y"
if(check == "Y")
{
    //Iterate through the list
    while(!Ilist->IsDone())
    {
        //current item in the list
        e = Ilist->Item();
        //check if entered ID is equal to the ID in the list
        if(id == e->getId())
        {
            //Remove employee object pointer from the list
            eList->Remove(e);
            //delete the employee object pointer
            delete e;
            break;
        }
        //point to next item in the list
        Ilist->Next();
    }
}

```

```

}
else{
    cout << endl;
    cout << "Employee Record Not Deleted" << endl;
}

```

i. Displaying individual paychecks and total paycheck using Apply function

```

    cout << ">-----< Employee >-----<" << endl << endl;
    cout << "Order #" << order << endl;
    cout << "Employee ID: " << e->getId() << " || Employee Name: " << e->getName() << endl;
    cout << "Working hours: " << e->getHoursWorked() << endl;
    cout << "PayCheck Amount: " << e->getpayCheck() << endl << endl;
    order++;

```

j. Function to get personal paycheck and total pay check

```

//iterate through the list
while(!Ilist->IsDone())
{
    //current item in the list
    e = Ilist->Item();
    //append employee object pointer to the list
    ePayList->Append(e);
    //point to net item in the list
    Ilist->Next();
}
//create list iterator for paycheck list
ListIterator<Employee*> *IlistPay = new ListIterator<Employee*>(ePayList);
//iterate through the list
while(!IlistPay->IsDone())
{
    //current item in the list
    e = IlistPay->Item();
    //append employee object pointer to the list
    totalPayCheck = totalPayCheck + e->getpayCheck();
    //point to net item in the list
    IlistPay->Next();
}
//display the paycheck amount in decreasing order of paycheck

```

```
ePayList->Apply(printShowCheck);
cout << endl;
cout << "Total Amount of PayCheck is: " << totalPayCheck << endl << endl;
```

k. Saving employee details to the file

```
//iterate through the list
while(!Ilist->IsDone())
{
    //current item in the list
    e = Ilist->Item();
    //output the employee record to the employee.dat file
    myfile << e->getId() << " " << e->getName() << " " << e->getDept() << " "
<< e->getpos() << " " << e->gethrate() << endl;
    //point to next item in the list
    Ilist->Next();
}
//close the file
myfile.close();
```

l. Selecting operations to be performed on employee record

```
int option=0;
while (option != 7)
{
    cout << "Select 1 to enter new record" << endl;
    cout << "Select 2 to display all the Employees" << endl;
    cout << "Select 3 to search an Employee" << endl;
    cout << "Select 4 to Update employee details" << endl;
    cout << "Select 5 to Delete an Employee information" << endl;
    cout << "Select 6 to get the weekly personal paychecks and total paycheck
amount" << endl;
    cout << "Select 7 to quit the program" << endl << endl;
    cout << "Select any one option" << endl;
    cin >> option;
    //switch statement select menu
    switch (option) {
        //function to enter new record
        case 1: enterNewRecord();
            break;
        //function to display all the employee records
        case 2: eList->Apply(displayEmployees);
```



```

        break;
        //function to search employee record
case 3: SearchEmployee();
        break;
        //function to update employee details
case 4: UpdateDetails();
        break;
        //function to delete employee record
case 5: DeleteEmployee();
        break;
        //function to display individual paychecks and total paycheck amount
case 6: ShowCheck();
        break;
        //function to save to the file
case 7: saveToFile();
        exit(0);

```

Testing

How to run the Test

1. Copy the nachos folder from your local machine to the server.
2. Copy the header file Employee.h and class file Employee.cc to the location /nachos/code/threads/
3. Copy the modified Threadtest.cc from the local machine to the server to the location /nachos/code/threads/
4. Navigate to the directory nachos/code/build.linux
5. Execute the below commands
 - a. make clean
 - b. make depend
 - c. make nachos
 - d. ./nachos -K
6. To break out from the program, Select 7 from the Menu.

Added Files:

Employee.cc, Employee.h and employee.dat to the location /nachos/code/threads/

Modified Files:

Threadtest.cc in the location /nachos/code.threads/

Makefile

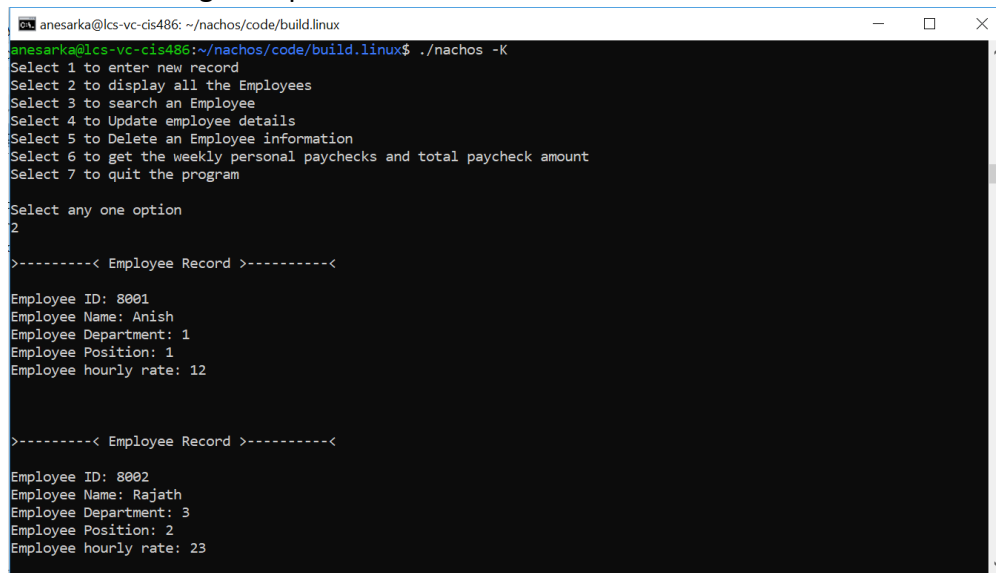
```
THREAD_H = ../threads/alarm.h\  
          ../threads/kernel.h\  
          ../threads/main.h\  
          ../threads/scheduler.h\  
          ../threads/switch.h\  
          ../threads/synch.h\  
          ../threads/synchlist.h\  
          ../threads/thread.h\  
          ../threads/employee.h
```

```
THREAD_C = ../threads/alarm.cc\  
          ../threads/kernel.cc\  
          ../threads/main.cc\  
          ../threads/scheduler.cc\  
          ../threads/synch.cc\  
          ../threads/synchlist.cc\  
          ../threads/thread.cc\  
          ../threads/threadtest.cc\  
          ../threads/employee.cc
```

```
THREAD_O = alarm.o kernel.o main.o scheduler.o synch.o thread.o threadtest.o  
employee.o
```

TestCases and Output

1. Display existing employees from the file by selecting option 2. The output is in sorted order according to department



```
anesarka@lcs-vc-cis486: ~/nachos/code/build.linux  
anesarka@lcs-vc-cis486:~/nachos/code/build.linux$ ./nachos -K  
Select 1 to enter new record  
Select 2 to display all the Employees  
Select 3 to search an Employee  
Select 4 to Update employee details  
Select 5 to Delete an Employee information  
Select 6 to get the weekly personal paychecks and total paycheck amount  
Select 7 to quit the program  
  
Select any one option  
2  
  
>-----< Employee Record >-----<  
  
Employee ID: 8001  
Employee Name: Anish  
Employee Department: 1  
Employee Position: 1  
Employee hourly rate: 12  
  
>-----< Employee Record >-----<  
  
Employee ID: 8002  
Employee Name: Rajath  
Employee Department: 3  
Employee Position: 2  
Employee hourly rate: 23
```

2. Entering new record of the employee data

- a. Select option 1 from the menu
- b. Type the name
- c. Type position
- d. Type department
- e. Type hourly rate

```
anesarka@lcs-vc-cis486: ~/nachos/code/build.linux
```

```
anesarka@lcs-vc-cis486:~/nachos/code/build.linux$ ./nachos -K
Select 1 to enter new record
Select 2 to display all the Employees
Select 3 to search an Employee
Select 4 to Update employee details
Select 5 to Delete an Employee information
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program

Select any one option
1
Enter Employee Name
Anish
Enter Position
1
Enter Department
1
Enter hourly rate
12
Select 1 to enter new record
Select 2 to display all the Employees
Select 3 to search an Employee
Select 4 to Update employee details
Select 5 to Delete an Employee information
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program

Select any one option
2

>-----< Employee Record >-----<

Employee ID: 8001
Employee Name: Anish
Employee Department: 1
Employee Position: 1
Employee hourly rate: 12
```

3. Display all employees - > Select option 2 from the Menu

The employees are sorted according to department first and then according to position when departments are same.

```
anesarka@lcs-vc-cis486: ~/nachos/code/build.linux
```

```
anesarka@lcs-vc-cis486:~/nachos/code/build.linux$ ./nachos -K
Select 1 to enter new record
Select 2 to display all the Employees
Select 3 to search an Employee
Select 4 to Update employee details
Select 5 to Delete an Employee information
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program
```

```
Select any one option
```

```
2
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8001
Employee Name: Anish
Employee Department: 1
Employee Position: 1
Employee hourly rate: 12
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8003
Employee Name: Karan
Employee Department: 2
Employee Position: 2
Employee hourly rate: 13
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8002
Employee Name: Rajath
Employee Department: 3
Employee Position: 4
Employee hourly rate: 23
```

4. Search an employee

- a. Select 3 to search an employee
- b. Select 1 to search by Employee ID and 2 to search by employee department
- c. If "1" Enter the employee ID. If "2", Enter the employee department.

```
anesarka@lcs-vc-cis486: ~/nachos/code/build.linux
```

```
anesarka@lcs-vc-cis486:~/nachos/code/build.linux$ ./nachos -K
```

```
Select 1 to enter new record  
Select 2 to display all the Employees  
Select 3 to search an Employee  
Select 4 to Update employee details  
Select 5 to Delete an Employee information  
Select 6 to get the weekly personal paychecks and total paycheck amount  
Select 7 to quit the program
```

```
Select any one option
```

```
3
```

```
To Search:
```

```
Select 1 to search by ID
```

```
Select 2 to search by Department
```

```
1
```

```
Enter Employee ID
```

```
8002
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8002
```

```
Employee Name: Rajath
```

```
Employee Department: 3
```

```
Employee Position: 2
```

```
Employee hourly rate: 23
```

```
Select 1 to enter new record
```

```
Select 2 to display all the Employees
```

```
Select 3 to search an Employee
```

```
Select 4 to Update employee details
```

```
Select 5 to Delete an Employee information
```

```
Select 6 to get the weekly personal paychecks and total paycheck amount
```

```
Select 7 to quit the program
```

```
anesarka@lcs-vc-cis486: ~/nachos/code/build.linux
```

```
anesarka@lcs-vc-cis486:~/nachos/code/build.linux$ ./nachos -K
```

```
Select 1 to enter new record
```

```
Select 2 to display all the Employees
```

```
Select 3 to search an Employee
```

```
Select 4 to Update employee details
```

```
Select 5 to Delete an Employee information
```

```
Select 6 to get the weekly personal paychecks and total paycheck amount
```

```
Select 7 to quit the program
```

```
Select any one option
```

```
3
```

```
To Search:
```

```
Select 1 to search by ID
```

```
Select 2 to search by Department
```

```
2
```

```
Enter Department number
```

```
1
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8001
```

```
Employee Name: Anish
```

```
Employee Department: 1
```

```
Employee Position: 1
```

```
Employee hourly rate: 12
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8003
```

```
Employee Name: Anish
```

```
Employee Department: 1
```

```
Employee Position: 1
```

```
Employee hourly rate: 12
```

5. Update Employee details

a. Select 4 to update employee details

- i. Select 1 to enter new name, 2 to enter new department, 3 to enter new position, 4 to enter new hourly rate, 5 to exit

```
anesarka@lcs-vc-cis486: ~/nuchos/code/build.linux
anesarka@lcs-vc-cis486:~/nuchos/code/build.linux$ ./nuchos -K
Select 1 to enter new record
Select 2 to display all the Employees
Select 3 to search an Employee
Select 4 to Update employee details
Select 5 to Delete an Employee information
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program
```

Select any one option

4

Enter Employee ID

8001

1. Update Name
2. Update Department
3. Update Position
4. Update hourly rate
5. Exit

Select Option

1

Enter New Name

Nishant

1. Update Name
2. Update Department
3. Update Position
4. Update hourly rate
5. Exit

Select Option

5

```
anesarka@lcs-vc-cis486: ~/nuchos/code/build.linux
```

Enter New Name

Nishant

1. Update Name
2. Update Department
3. Update Position
4. Update hourly rate
5. Exit

Select Option

5

```
Select 1 to enter new record
Select 2 to display all the Employees
Select 3 to search an Employee
Select 4 to Update employee details
Select 5 to Delete an Employee information
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program
```

Select any one option

2

>-----< Employee Record >-----<

```
Employee ID: 8001
Employee Name: Nishant
Employee Department: 1
Employee Position: 1
Employee hourly rate: 12
```

6. Delete Employee Record
 - a. Select 6 to delete employee record
 - b. Enter Employee ID
 - c. Select Y/N to confirm to delete.

```
C:\> anesarka@lcs-vc-cis486: ~/nachos/code/build.linux
```

```
Select 5 to Delete an Employee information
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program
```

```
Select any one option
```

```
5
```

```
Enter the employee ID
```

```
8001
```

```
Confirm Y/N
```

```
Y
```

```
Select 1 to enter new record
```

```
Select 2 to display all the Employees
```

```
Select 3 to search an Employee
```

```
Select 4 to Update employee details
```

```
Select 5 to Delete an Employee information
```

```
Select 6 to get the weekly personal paychecks and total paycheck amount
```

```
Select 7 to quit the program
```

```
Select any one option
```

```
2
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8003
```

```
Employee Name: Anish
```

```
Employee Department: 1
```

```
Employee Position: 1
```

```
Employee hourly rate: 12
```

```
>-----< Employee Record >-----<
```

```
Employee ID: 8002
```

```
Employee Name: Rajath
```

```
Employee Department: 3
```

```
Employee Position: 2
```

```
Employee hourly rate: 23
```

```
Select 1 to enter new record
```

7. Select 6 to display personal employee paychecks in descending order and total paycheck amount of all the employees.

```
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program
```

```
Select any one option
```

```
6
```

```
>-----< Employee >-----<
```

```
Order #1
```

```
Employee ID: 8002 || Employee Name: Rajath
```

```
Working hours: 29
```

```
PayCheck Amount: 667
```

```
>-----< Employee >-----<
```

```
Order #2
```

```
Employee ID: 8003 || Employee Name: Anish
```

```
Working hours: 24
```

```
PayCheck Amount: 288
```

```
>-----< Employee >-----<
```

```
Order #3
```

```
Employee ID: 8001 || Employee Name: Anish
```

```
Working hours: 21
```

```
PayCheck Amount: 252
```

```
Total Amount of PayCheck is: 1207
```

```
Select 1 to enter new record
```

```
Select 2 to display all the Employees
```

```
Select 3 to search an Employee
```

```
Select 4 to Update employee details
```

```
Select 5 to Delete an Employee information
```

```
Select 6 to get the weekly personal paychecks and total paycheck amount
```

```
Select 7 to quit the program
```


8. Select 7 to save to file and exit the program

```
anesarka@lcs-vc-cis486: ~/nachos/code/build.linux
>-----< Employee Record >-----<

Employee ID: 8001
Employee Name: Anish
Employee Department: 1
Employee Position: 1
Employee hourly rate: 12

>-----< Employee Record >-----<


Employee ID: 8003
Employee Name: Anish
Employee Department: 2
Employee Position: 4
Employee hourly rate: 12

>-----< Employee Record >-----<

Employee ID: 8002
Employee Name: Rajath
Employee Department: 3
Employee Position: 2
Employee hourly rate: 23

Select 1 to enter new record
Select 2 to display all the Employees
Select 3 to search an Employee
Select 4 to Update employee details
Select 5 to Delete an Employee information
Select 6 to get the weekly personal paychecks and total paycheck amount
Select 7 to quit the program

Select any one option
7
Saving Employee Data to employee.dat file
anesarka@lcs-vc-cis486:~/nachos/code/build.linux$
```

 /home/anesarka/nachos/code/threads/employee.dat - anesarka@lcs-vc-cis486.syr.edu -

```

8001 Anish 1 1 12
8003 Anish 2 4 12
8002 Rajath 3 2 23
```