CIS 657: Operating Systems

Lab Assignment 2

Date: 2/15/2019, Spring 2019

Submitted by,
Lakshmi Harshini Kuchibhotla
SUID – 230997383
SUEmail – lkuchibh@syr.edu

CIS657 Spring 2019

Assignment Disclosure Form

Assignment #: 1

Name: Kuchibhotla Lakshmi Harshini

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

   If Yes, please give the details.

Yes

Shared thoughts and ideas with Bhargav Rahul

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.

No

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

Signature:___K.L.Harshini._________ Date : 2/15/2019

Design for each requirement:

Requirement 1- Create new record

Implemented using set functions. User is given an access to input all the info of the employee (name, position, department, payrate) using the get functions. And these inputs are set to the employee record using set functions. Provided options for user to choose the position and department from given choices. All the created records are added to the List using functions from list.h.

Requirement 2- Display Records

Until the list is empty, the records of all the employees is printed. The records are sorted first with department lexicographically using SortedList if there is a match, it sorts with position and the sorted database is printed

Requirement 3- Search Employee

Implemented in 2 functions by search using id and search using department. As the id is unique, when we search only one record is displayed if id exists in the database. If searched using department, it recursively searches the database for the match of the department entered using ListIterator. It prints all the employees records with the corresponding department.

Requirement 4- Update Employee record

Employee info is updated by checking the ID of the employee. Then the info is updated with the new user given values. Two types of update are implemented – Update all the info, Update single info. In update all info, every value is taken from the user and in single update, only the respective field is updated by user given value.

Requirement 5- Delete employee record

Employee record is deleted using the unique Id. Once the id is entered, it asks for the confirmation message to delete the record. User need to confirm if he really wants to delete or not. It is deleted from the database using the Remove().

Requirement 6- Compute Pay Check after jobs scheduled

Pay check is calculated using the number of hours generated randomly and the pay rate given by the user. Sort the employee records using SortedList for pay check in descending order and print all the records using ListIterator. Finally, the total amount of all the paychecks is also displayed.

Requirement 7-

It just exits the kernel program but before that it writebacks the employee records to the file using fstream methods. If employee.dat exists, it overwrites the file, else it creates the file and stores the records. Also deleted the pointers.

Implementation of the solution-Code:

employee.h

```
/* This class contains the declarations for the getter and setter functions
needed for the employee record.
all the member variables are declared in private and the member functions are
declared in public*/
//Autor: Lakshmi Harshini Kuchibhotla, SU ID: 230997383, SU email:
lkuchibh@syr.edu
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <string>
#include <stdlib.h>

class Employee
{
    public:
        Employee(std::string name, int id, std::string pos, std::string dept, int
payrate); //constructor to print basic employee info
        bool operator == (const Employee &emp);
        Employee(std::string name, int id, std::string pos, std::string dept, int
payrate, int noofhours, int salary); //constructor with all the inputs
        int getID();
        std::string getname();
        void setname(std::string name);
        std::string getposition();
        void setposition(std::string pos);
        std::string getdepartment();
        void setdepartment(std::string dept);
        int getpayrate();
        void setpayrate(int payrate);
        int getnoofhours();
        int getsalarypaycheck();

    private:
        int id;
        std::string name;
        std::string pos;
        std::string dept;
        int payrate;
        int noofhours;
        int salary;
};
```

```
#endif //#endif directive is to end the preceeding #if.
```

employee.cc

```
/*This file contains the implementations of the member functions declared in
employee.h file*/
//Autor: Lakshmi Harshini Kuchibhotla, SU ID: 230997383, SU email:
lkuchibh@syr.edu
#include "employee.h"
#include <stdlib.h>

//Implementation for the constructor
Employee::Employee(std::string name, int id, std::string pos, std::string dept,
int payrate)
{
    this->id = id;
    this->name = name;
    this->pos = pos;
    this->dept = dept;
    this->payrate = payrate;
    this->noofhours = (rand() % 20) + 20;
    this->salary = noofhours*payrate;
}

//operator overloading - done to use in list class
bool
Employee::operator == (const Employee & emp)
{
    if(this->id == emp.id)
    {
        return true;
    }
    else
    {
        return false;
    }
}

//constructor for Employee class which produces salary and the number of hours
worked
Employee::Employee(std::string name, int id, std::string pos, std::string dept,
int payrate, int noofhours, int salary)
{
    this->id = id;
    this->name = name;
    this->pos = pos;
    this->dept = dept;
    this->payrate = payrate;
    this->noofhours = noofhours;
    this->salary = salary;
}

//function to get the employee ID - returns id;
int
Employee::getID()
{
    return id;
}

//function to get the employee name - returns a string
std::string
Employee::getname()
{
    return name;
}

//function to get the employee Position - returns a string
std::string
Employee::getposition()
{
    return pos;
}

//function to get the employee department - returns a string
std::string
Employee::getdepartment()
{
    return dept;
}

//function to get the pay rate - returns integer
int
Employee::getpayrate()
{
    return payrate;
}

//function to get the hours worked - returns an integer
int
Employee::getnoofhours()
{
    return this->noofhours;
```

```
}

//function to get the paycheck for the week - returns a number
int
Employee::getsalarypaycheck()
{
    return this->salary;
}

//function to set the employee name to the given input
void
Employee::setname(std::string name)
{
    this->name = name;
}

//function to set the employee position to the user given input
void
Employee::setposition(std::string pos)
{
    this->pos = pos;
}

//function to set the employee department to the user given input
void
Employee::setdepartment(std::string dept)
{
    this->dept = dept;
}

//function to set the pay rate to the user given input
void
Employee::setpayrate(int payrate)
{
    this->payrate = payrate;
}
```

threadtest.cc

```
/*This file contains the interactive process between the nachos and the employee
files.
When nachos -K is run, ThreadTest() gets the control and the Menu is displayed
for the user */
//Autor: Lakshmi Harshini Kuchibhotla, SU ID: 230997383, SU email:
lkuchibh@syr.edu
```

```
#include "main.h"
#include "employee.h"
#include <string>
#include "list.h"
#include <iostream>
#include <sstream>
#include <fstream>

//function converts the integer to the string value.
std::string
Convert_IntTostring(int integer)
{
    std::ostringstream outputss;
    outputss << integer;
    return outputss.str();
}

//function which compares the position of the employees and returns the
comparision result
int
SortPosition(Employee* e1,Employee* e2)
{
    int result = (*e1).getposition().compare((*e2).getposition());
    return result;
}

//function which compares the department of the employees and returns the
comparision result
int
SortDepartment(Employee* e1,Employee* e2)
{
    int result = (*e1).getdepartment().compare((*e2).getdepartment());
    return result;
}

//function sorts the employees using department and position sort.
int
SortEmployee(Employee* e1,Employee* e2)
{
    int posresult,deptresult;
    posresult = SortPosition(e1, e2);
    deptresult = SortDepartment(e1, e2);
    if(deptresult != 0)
    {
        return deptresult;
    }
    return posresult;
}

//to print the database - function prints the records of all the employees
entried
void
DisplayRecords(Employee* emp)
{
    cout<<" Name: " << (*emp).getname() << "   |   ID: " << (*emp).getID() << "
|   Position: " << (*emp).getposition()
        << "   |   Department: " << (*emp).getdepartment() << "   |   Payrate: "
<< (*emp).getpayrate() << endl;
}

//search employee implementation - function searches the employee using the
department name and returns true if dept found
bool
Search_usingDept(SortedList<Employee*> empList, std::string search_dept)
{
    bool deptfound = false;
    ListIterator<Employee*> *itr = new ListIterator<Employee*> (&empList);
    for(; !itr->IsDone(); itr->Next())
    {
        std::string current_dept = itr->Item()->getdepartment();
        if(current_dept == search_dept)
        {
            DisplayRecords(itr->Item());
            deptfound = true;
        }
    }
    return deptfound;
}

//Searches the employee info using the employee ID - returns true if the ID is
found
bool
Search_usingId(SortedList<Employee*> empList, std::string search_id)
{
    ListIterator<Employee*> *itr = new ListIterator<Employee*> (&empList);
    for(; !itr->IsDone(); itr->Next())
    {
        std::string current_id = Convert_IntTostring(itr->Item()->getID());
        if(current_id == search_id)
        {
            DisplayRecords(itr->Item());
            return true;
        }
    }
    return 0;
}

//Prints the Menu for the Employee Management Database
void
DisplayMenu()
{
    cout<< "1. Enter new record" << endl << "2. Display all employees" << endl
        << "3. Search employee(s)" << endl << "4. Update employee info" << endl
        << "5. Delete employee info" << endl << "6. Schedule weekly jobs" << endl
        << "7. Exit" << endl;
}

//function displays the employees records that are searched using ID or
department
void
Display_EmployeeSearched(SortedList<Employee*> empList)
{
    std::string search_str;
    cout<< endl <<"Enter either Department or Id to search: ";
    cin.clear();
    cin.ignore(1000,'\n');
    std::getline(std::cin, search_str);

    if(!Search_usingId(empList, search_str))
    {
        if(!Search_usingDept(empList, search_str))
        {
            cout<< "Invalid ID or Department"<<endl;
            cout<< "***** No records found *****"<<endl;
        }
    }
}

//function updates the employees info to the new user given inputs by using the
unique id.
void
Update_EmployeeInfo(SortedList<Employee*> empList)
{
    std::string update_id;
    bool Id_found = false;
    cout<< endl <<"Enter Employee Id of your choice to update his info: ";
    cin>> update_id;

    ListIterator<Employee*> *itr = new ListIterator<Employee*> (&empList);
    for(; !itr->IsDone(); itr->Next())
    {
        std::string current_id = Convert_IntTostring(itr->Item()->getID());
        if(update_id == current_id)
        {
            Id_found = true;
            int update_option, update_field;
            cout<< "Do you want to update all the fields or a particular field?"
<< endl;
            cout<< "Enter 1.Update all Fields   2. Particular Field :  ";
            cin>> update_option;
            if (update_option == 1)
            {
                std::string updated_name, updated_pos, updated_dept;
                int updated_payrate;
                cout<< "Enter the updated Name: ";
                cin>> updated_name;
                cout<< "Enter the updated Position(1/2/3/4/5/6: ) ";
                cin>> updated_pos;
                cout<< "Enter the updated Department(1/2/3/4): ";
                cin>> updated_dept;
                itr->Item()->setname(updated_name);
                itr->Item()->setposition(updated_pos);
                itr->Item()->setdepartment(updated_dept);

                cout<< "Enter the updated Payrate: ";
                cin>> updated_payrate;
                if(cin.fail())
                {
                    cout<< "Payrate must be a numerical value." << endl;
                    break;
                }
                itr->Item()->setpayrate(updated_payrate);
                break;
            }
            else if (update_option == 2)
            {
                cout<< "Which field you want to update?" << endl;
                cout<< "1.Name  2.Position  3.Department  4.Payrate: ";
                cin>> update_field;
```

```
switch(update_field)
{
    case 1:
    {
        std::string updated_name;
        cout<< "Enter the updated Name: ";
        cin>> updated_name;
        itr->Item()->setname(updated_name);
    }
    break;
    case 2:
    {
        std::string updated_pos;
        cout<< "Enter the updated Position: ";
        cin>> updated_pos;
        itr->Item()->setposition(updated_pos);
    }
    break;
    case 3:
    {
        std::string updated_dept;
        cout<< "Enter the updated Department: ";
        cin>> updated_dept;
        itr->Item()->setdepartment(updated_dept);
    }
    break;
    case 4:
    {
        int updated_payrate;
        cout<< "Enter the updated Payrate: ";
        cin>> updated_payrate;
        if(cin.fail())
        {
            cout<< "Payrate must be a numerical value." << endl;
            break;
        }
        itr->Item()->setpayrate(updated_payrate);
    }
    break;
    default:
    {
        cout<< "It is an invalid option" << endl;
    }
    break;
}
```

```
            }
            else
            {
                cout<< " ***** Invalid option ***** " << endl;
            }
        }
    }
    if(!Id_found)
    {
        cout<< "The requested Id does not exist." << endl << "***** Hence no
records are updated *****" << endl;
    }
}

//function deletes the corresponding employee records given the correct employee
ID
void
Delete_EmployeeInfo(SortedList<Employee*> &empList)
{
    std::string delete_id;
    bool Id_found = false;
    cout<< endl << "Enter an Employee ID to delete his/her record from the
database: ";
    cin>> delete_id;

    ListIterator<Employee*> *itr = new ListIterator<Employee*> (&empList);
    Employee* emp;
    for(; !itr->IsDone(); itr->Next())
    {
        std::string current_id = Convert_IntTostring(itr->Item()->getID());
        if(delete_id == current_id)
        {
            Id_found =  true;
            std::string sure;
            cout<< endl << "This is a confirmation to delete the requested Id."<<
endl;
            cout<< "Enter (Y/y) to delete: ";
            cin>> sure;

            if(sure == "Y" || sure == "y")
            {
                emp = itr->Item();
                empList.Remove(emp);
                cout<<endl<<"The requested employee record is deleted";
            }
            else
            {
                cout<< endl <<"You chose not to delete any record"<< endl;
            }

        }
    }
    if(!Id_found)
    {
        cout<< "The requested Id does not exist." << endl << "***** Hence no 
records are deleted *****" << endl;
    }
}

//compares the paycheck amounts of the employees and returns them in descending 
order
int
SalaryCompare(Employee* e1, Employee* e2)
{
    if(e1->getsalarypaycheck() < e2->getsalarypaycheck())
    {
        return 1;
    }
    return -1;
}

/*function generates the paychecks of all the employees after the job is 
scheduled
for the week using the number of hours worked and the payrate.
It also prints the total amount of the paycheck of all the employees*/
void
PayCheckAmount(SortedList<Employee*> empList)
{
    SortedList<Employee*> employeeList(SalaryCompare);
    ListIterator<Employee*> *itr = new ListIterator<Employee*> (&empList);
    for(; !itr->IsDone(); itr->Next())
    {
        employeeList.Insert(itr->Item());
    }

    int finalamount = 0, Order_no = 1;

    ListIterator<Employee*> *itr1 = new ListIterator<Employee*> (&employeeList);
    for(; !itr1->IsDone(); itr1->Next())
    {
        int amount = itr1->Item()->getsalarypaycheck();
        finalamount += amount;
        cout<< "Order " << Order_no++ <<" |  ID: "<< itr1->Item()->getID() <<"
|   Name: " << itr1->Item()->getname()
            << "    |   NumberofHours: " << itr1->Item()->getnoofhours() << "    |
PayCheck: " << amount
            << "   |   Payrate: " << itr1->Item()->getpayrate() << endl;
    }
    cout<< "The total amount of pay check: " << finalamount <<endl;
}

//create employee - takes the input from the user for all the fields to create
the employee record
Employee* EmployeeRecord(int id)
{
    std::string empname, empdept, emppos;
    int empid, dept, pos, emppayrate;
    cout<< endl <<"Enter Name: ";
    cin>> empname;
    empid = id;
    bool poschoice = true, deptchoice = true;
    while(poschoice)
    {
        cout<<endl<<"Choose from the mentioned Positions";
        cout<<endl<< "1. Intern | 2. Entry Level | 3. Associate | 4. Senior | 5.
Lead | 6.Manager " << endl;
        cout<< "Enter Position(1/2/3/4/5/6): ";
        cin>> pos;
        if (pos == 1)
        {
            emppos = "Intern";
            poschoice = false;
        }
        else if(pos == 2)
        {
            emppos = "Entry Level";
            poschoice = false;
        }
        else if (pos == 3)
        {
            emppos = "Associate";
            poschoice = false;
        }
        else if (pos == 4)
        {
            emppos = "Senior";
            poschoice = false;
        }
        else if (pos == 5)
        {
            emppos = "Lead";
            poschoice = false;
        }
        else if (pos == 6)
        {
            emppos = "Manager";
            poschoice = false;
        }
        else
        {
            cout<<endl<< "Invalid entry! Enter a number from the given
positions.";
            poschoice = true;
        }
        cin.clear();
        cin.ignore(10000,'\n');
    }
    while(deptchoice)
    {
        cout<<endl<<"Choose from the mentioned Departments";
        cout<<"1. Administrative | 2. HR | 3. Software Development | 4. Business
Analytics " << endl;
        cout<<endl<< "Enter Department(1/2/3/4): ";
        cin>> dept;
        if (dept == 1)
        {
            empdept = "Administrative";
            deptchoice = false;
        }
        else if (dept == 2)
        {
            empdept = "HR";
            deptchoice = false;
        }
        else if (dept == 3)
        {
            empdept = "Software Development";
            deptchoice = false;
        }
        else if (dept == 4)
        {
            empdept = "Business Analytics";
            deptchoice = false;
        }
        else
        {
            cout<< "Invalid entry! Enter a number from the given departments.";
            deptchoice = true;
        }
        cin.clear();
        cin.ignore(10000,'\n');
    }

    cout<< "Enter the payrate for the employee: ";

    cin>> emppayrate;
    cin.clear();
    cin.ignore(10000,'\n');
    Employee *emp = new Employee(empname, empid, emppos, empdept, emppayrate);
    return emp;
}

//function copies all the employee records and stores it to the employee.dat
before exiting the program
void
ConsoleToFile(SortedList<Employee*> empList)
{
    ofstream myfile;
    myfile.open("employee.dat");

    ListIterator<Employee*> *itr = new ListIterator<Employee*> (&empList);
    for(; !itr->IsDone(); itr->Next())
    {
        myfile<< itr->Item()->getname() << "," << itr->Item()->getID() << "," <<
itr->Item()->getposition()
            << "," << itr->Item()->getdepartment() << "," << itr->Item()-
>getpayrate() << ","
            << itr->Item()->getnoofhours() << "," << itr->Item()-
>getsalarypaycheck() << "," <<endl;

    }
    myfile.close();
    while(!empList.IsEmpty())
    {
        Employee* emp = empList.RemoveFront();
        delete emp;
    }
}

//ThreadTest() gets the control and calls all the necessary functions in a
corresponding sequence.
void
ThreadTest()
{
    int start_id = 8001;
    int next_id = start_id;

    SortedList<Employee*> employeeList(SortEmployee);
    ifstream filename("employee.dat");
    if(filename)
    {
        std::string database;
        while(std::getline(filename, database))
        {
            std::string delimiter = ",";
            int id, payrate, noofhours, salary, i=0;
            std::string name, pos, dept;
            size_t s = 0;
            std::string fields[7], field;

            while((s = database.find(delimiter)) != std::string::npos)
            {
                field = database.substr(0,s);
                fields[i] = field;
                database.erase(0, s+delimiter.length());
                i++;
            }

            name = fields[0];
            id = atoi(fields[1].c_str());
            pos = fields[2];
            dept = fields[3];
            payrate = atoi(fields[4].c_str());
            noofhours = atoi(fields[5].c_str());
            salary = atoi(fields[6].c_str());

            if(id > next_id)
            {
                next_id = id;
            }
            else
            {
                next_id = next_id;
            }
            Employee *emp = new Employee(name, id, pos, dept, payrate, noofhours,
salary);
            employeeList.Insert(emp);
        }
        next_id++;
    }

    cout<< "-------------------Employee Database Management Portal----------------
----------" << endl;
    bool exited = false;
    do
    {
        cout<< "    Menu: "<< endl;
        DisplayMenu();
        int option;
        cout<< "Enter your choice of operation: ";
        cin>> option;
        switch(option)
        {
            case 1:
            {
                Employee* emp = EmployeeRecord(next_id++);
                employeeList.Insert(emp);
            }
            break;
            case 2:
            {
                cout<< endl <<"***The employee details from the record***" <<
endl;
                employeeList.Apply(DisplayRecords);
                cout<< "     ----------------*****------------------     " <<
endl;
            }
            break;
            case 3:
            {
                Display_EmployeeSearched(employeeList);
                cout<< endl <<"     ----------------*****------------------     "
<< endl;
            }
            break;
            case 4:
            {
                Update_EmployeeInfo(employeeList);
                cout<< endl <<"     ----------------*****------------------     "
<< endl;
            }
            break;
            case 5:
            {
                Delete_EmployeeInfo(employeeList);
                cout<< endl <<"     ----------------*****------------------     "
<< endl;
            }
            break;
            case 6:
            {
                cout<<endl<< "***The employee records after scheduling jobs***"
<<endl;
                PayCheckAmount(employeeList);
                cout<< "     ----------------*****------------------     " <<
endl;
            }
            break;
            case 7:
            {
                ConsoleToFile(employeeList);
                exited = true;
            }
            cout<< endl << "------*****-------Exiting-------*****--------" <<
endl;
            delete kernel;
            exit(0);
            break;
            default:
                cout<< "Not a valid option" << endl;
                cin.clear();
                cin.ignore(10000,'\n');
            break;
        }

    } while (!exited);

}
```

Finally, in code/build.linux/Makefile, add ../threads/employee.h\ in THREAD_H, ../threads/employee.cc\ in THREAD_C, employee.o in THREAD_O.

Testing

1.Connect to the VM.

2.Enter make nachos in nachos/code/build.linux.

3.Enter ./nachos -K

It displays:

```
-------------------Employee Database Management Portal-------------------------

    Menu:

1. Enter new record

2. Display all employees

3. Search employee(s)

4. Update employee info

5. Delete employee info

6. Schedule weekly jobs

7. Exit

Enter your choice of operation:
```

Choose option 1 to create a new record if the database is empty or add a new record to the database.

-Enter the name, position, department and pay rate. ID is automatically generated by the system. Payrate should necessarily be int value.

Choose Option 2 to print all the existing employee records.

-If no records are there in the database, it prints the error message.

Choose Option 3 to Search the employees from the database and display the searched records.

-Search works using ID or Department.

If you enter ID, which is unique, the corresponding employee info is displayed. If ID not exists, it prints an error message.

If you enter Department, all the employee records corresponding to the department will be displayed. If invalid department, prints error message.

Choose Option 4 to update the corresponding employee info using the unique ID.

-Enter the unique ID of the employee to update his info. You get two options 1. Update all field 2. Update particular single field. Provide corresponding inputs (name, pos, dept, payrate) to update the record. Payrate should necessarily be int value.

Choose Option 5 to delete the employee record of the given employee ID.

-Enter the unique ID. It asks for a confirmation message. Enter Y/y, then the employee record is deleted from the database.

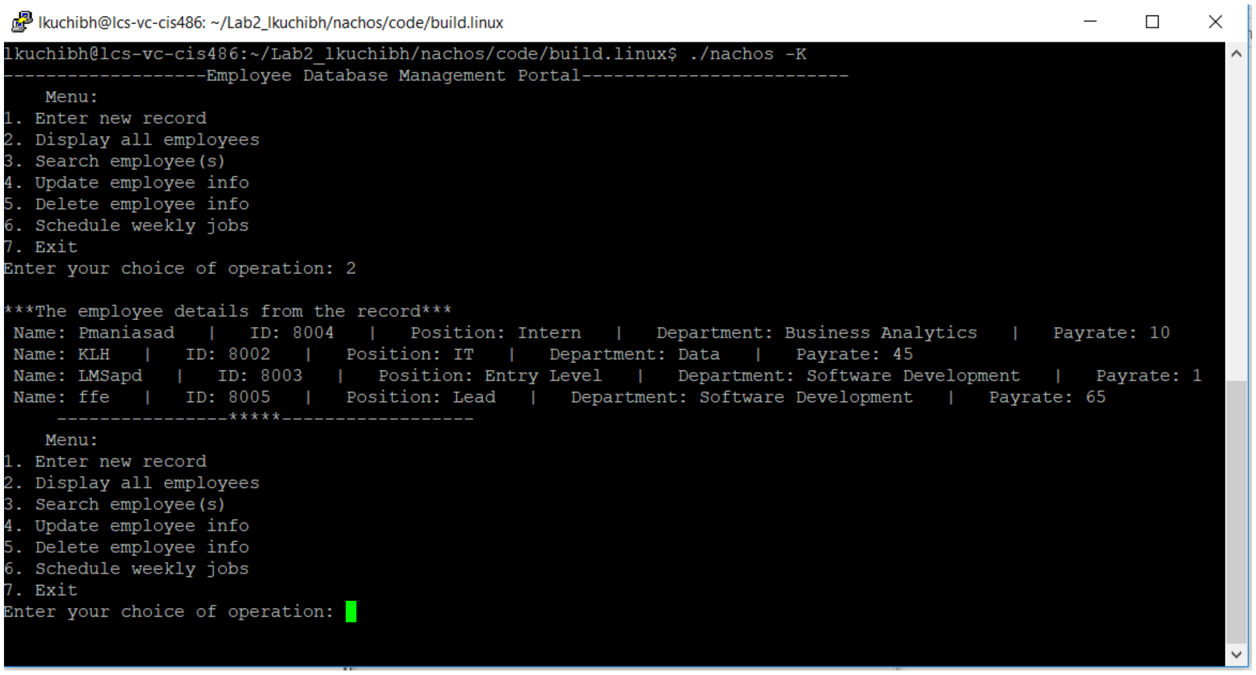
Choose Option 6 to calculate the pay check for the week after the job is scheduled.

It calculates the paycheck of all the employees in the database using the number of hours worked (randomly generated) and payrate. It then prints the employee records by sorting with pay check amount in descending order.

Choose Option 7 to store back the employee records to the file (employee.dat) and then it exits the program.

Output Snapshots:

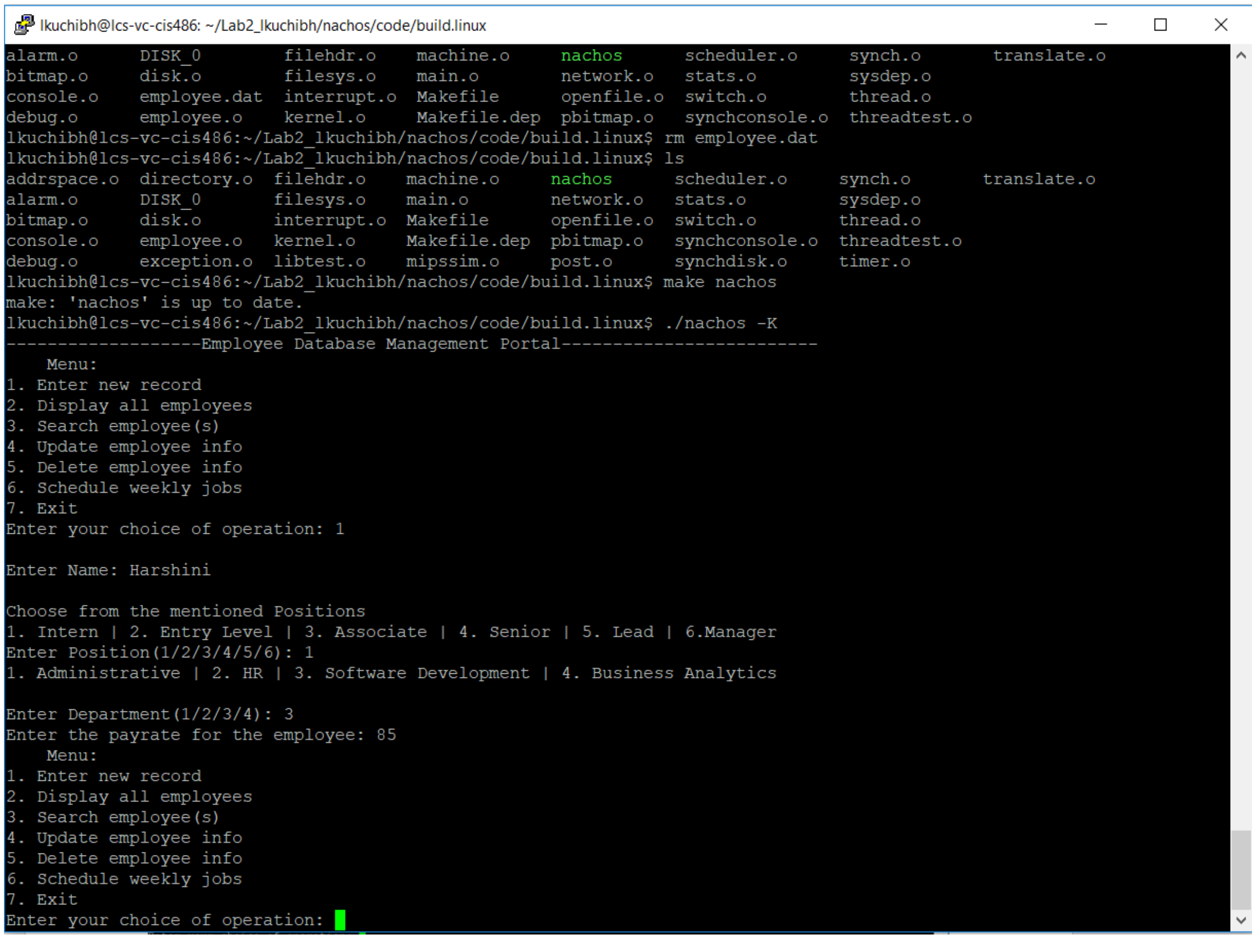
-Display records from the existing employee.dat



lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

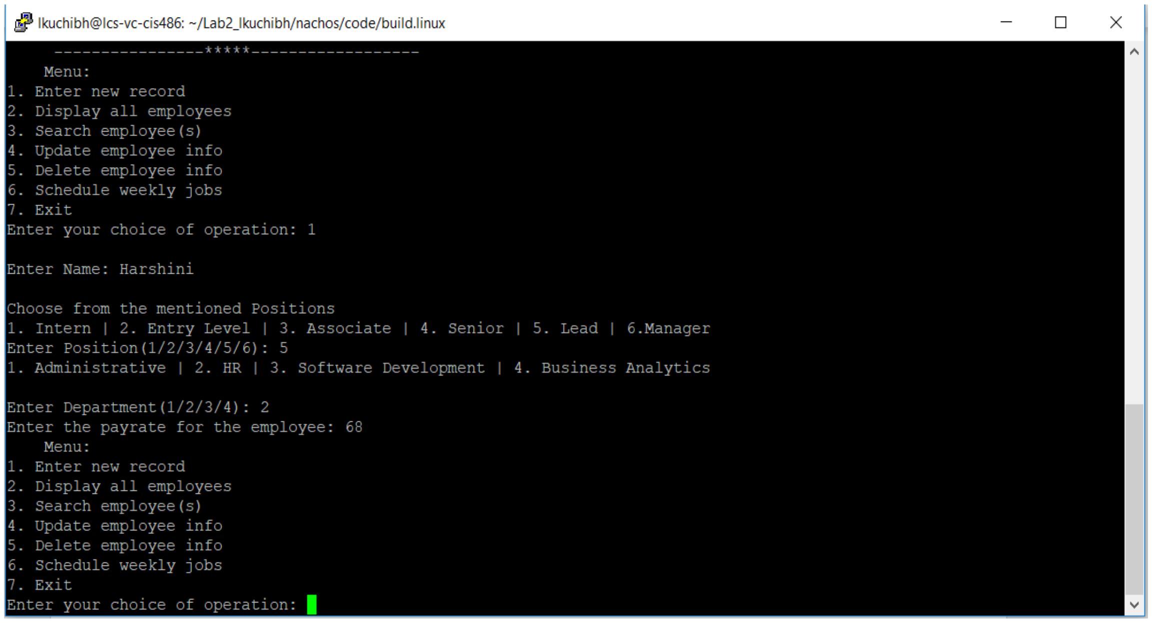
```
lkuchibh@lcs-vc-cis486:~/Lab2_lkuchibh/nachos/code/build.linux$ ./nachos -K
-------------------Employee Database Management Portal------------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: Pmaniasad   |   ID: 8004   |   Position: Intern   |   Department: Business Analytics   |   Payrate: 10
 Name: KLH   |   ID: 8002   |   Position: IT   |   Department: Data   |   Payrate: 45
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1
 Name: ffe   |   ID: 8005   |   Position: Lead   |   Department: Software Development   |   Payrate: 65
     ----------------*****-----------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

-Executing without a employee.dat file


lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

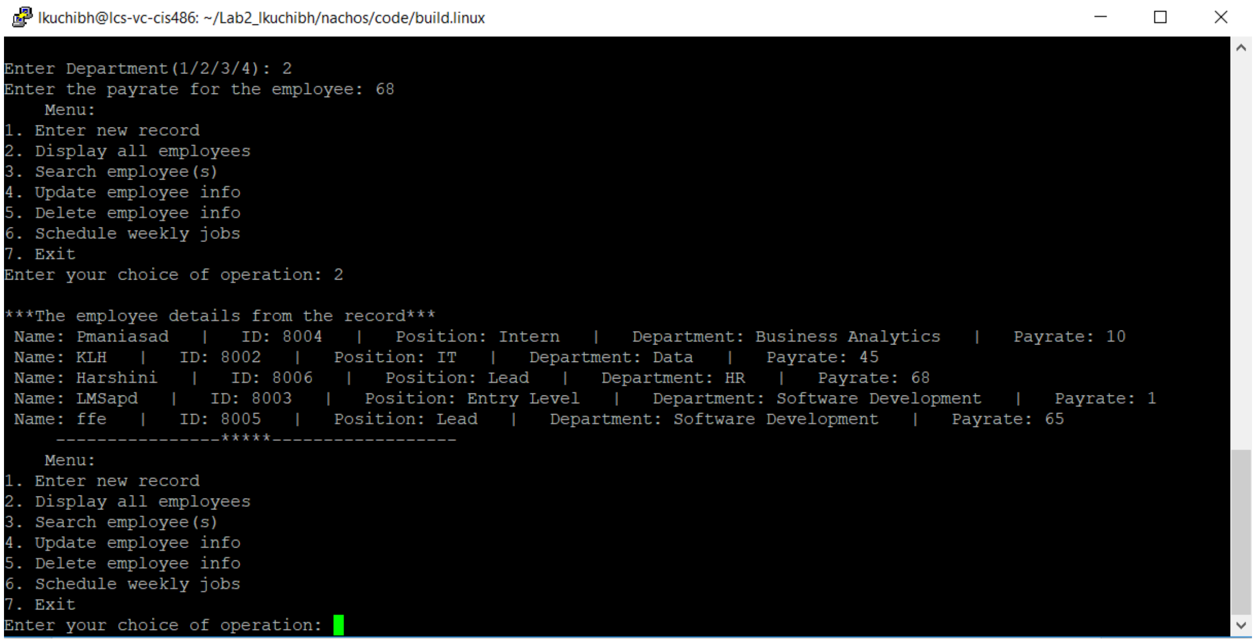
```
alarm.o      DISK_0       filehdr.o    machine.o    nachos      scheduler.o     synch.o       translate.o
bitmap.o     disk.o       filesys.o    main.o       network.o   stats.o         sysdep.o
console.o    employee.dat interrupt.o  Makefile     openfile.o  switch.o        thread.o
debug.o      employee.o   kernel.o     Makefile.dep pbitmap.o   synchconsole.o  threadtest.o
lkuchibh@lcs-vc-cis486:~/Lab2_lkuchibh/nachos/code/build.linux$ rm employee.dat
lkuchibh@lcs-vc-cis486:~/Lab2_lkuchibh/nachos/code/build.linux$ ls
addrspace.o  directory.o  filehdr.o    machine.o    nachos      scheduler.o     synch.o       translate.o
alarm.o      DISK_0       filesys.o    main.o       network.o   stats.o         sysdep.o
bitmap.o     disk.o       interrupt.o  Makefile     openfile.o  switch.o        thread.o
console.o    employee.o   kernel.o     Makefile.dep pbitmap.o   synchconsole.o  threadtest.o
debug.o      exception.o  libtest.o    mipssim.o    post.o      synchdisk.o     timer.o
lkuchibh@lcs-vc-cis486:~/Lab2_lkuchibh/nachos/code/build.linux$ make nachos
make: 'nachos' is up to date.
lkuchibh@lcs-vc-cis486:~/Lab2_lkuchibh/nachos/code/build.linux$ ./nachos -K
-------------------Employee Database Management Portal-------------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 1

Enter Name: Harshini

Choose from the mentioned Positions
1. Intern | 2. Entry Level | 3. Associate | 4. Senior | 5. Lead | 6.Manager
Enter Position(1/2/3/4/5/6): 1
1. Administrative | 2. HR | 3. Software Development | 4. Business Analytics

Enter Department(1/2/3/4): 3
Enter the payrate for the employee: 85
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

-Create a employee record


lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

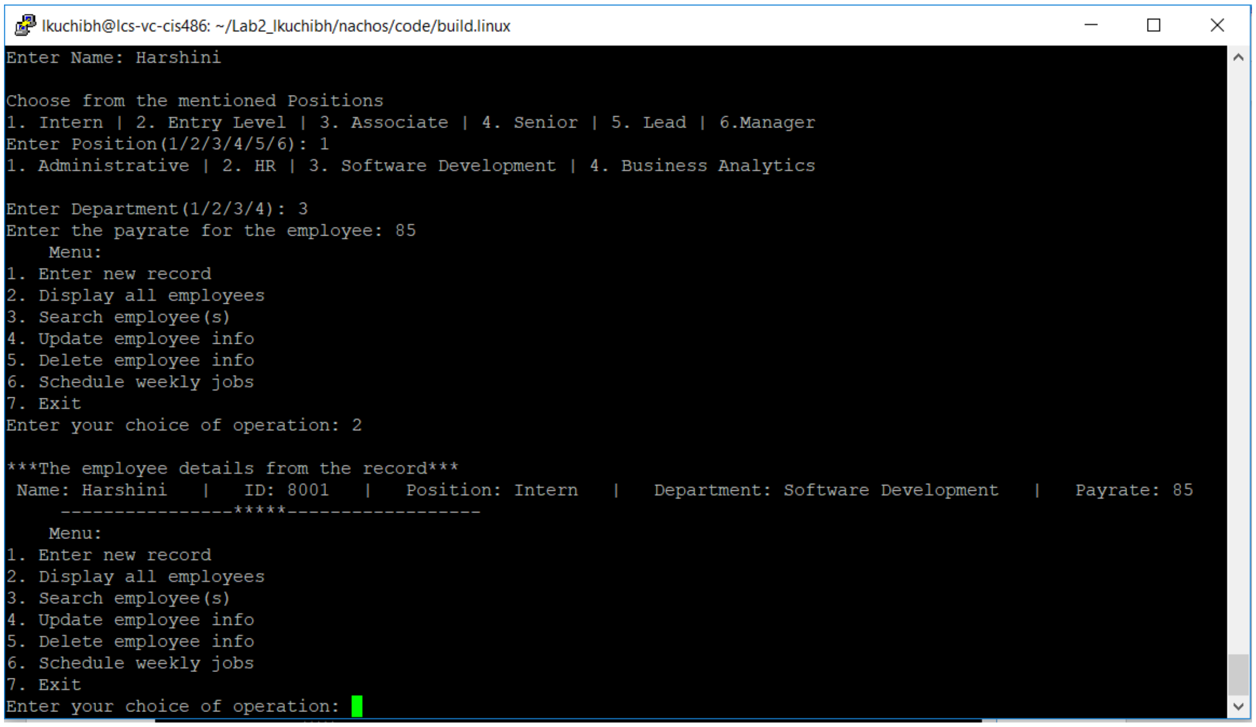
```
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 1

Enter Name: Harshini

Choose from the mentioned Positions
1. Intern | 2. Entry Level | 3. Associate | 4. Senior | 5. Lead | 6.Manager
Enter Position(1/2/3/4/5/6): 5
1. Administrative | 2. HR | 3. Software Development | 4. Business Analytics

Enter Department(1/2/3/4): 2
Enter the payrate for the employee: 68
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

-Display the Employee Records



lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

```
Enter Department(1/2/3/4): 2
Enter the payrate for the employee: 68
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: Pmaniasad    |    ID: 8004    |    Position: Intern    |    Department: Business Analytics    |    Payrate: 10
 Name: KLH    |    ID: 8002    |    Position: IT    |    Department: Data    |    Payrate: 45
 Name: Harshini    |    ID: 8006    |    Position: Lead    |    Department: HR    |    Payrate: 68
 Name: LMSapd    |    ID: 8003    |    Position: Entry Level    |    Department: Software Development    |    Payrate: 1
 Name: ffe    |    ID: 8005    |    Position: Lead    |    Department: Software Development    |    Payrate: 65
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

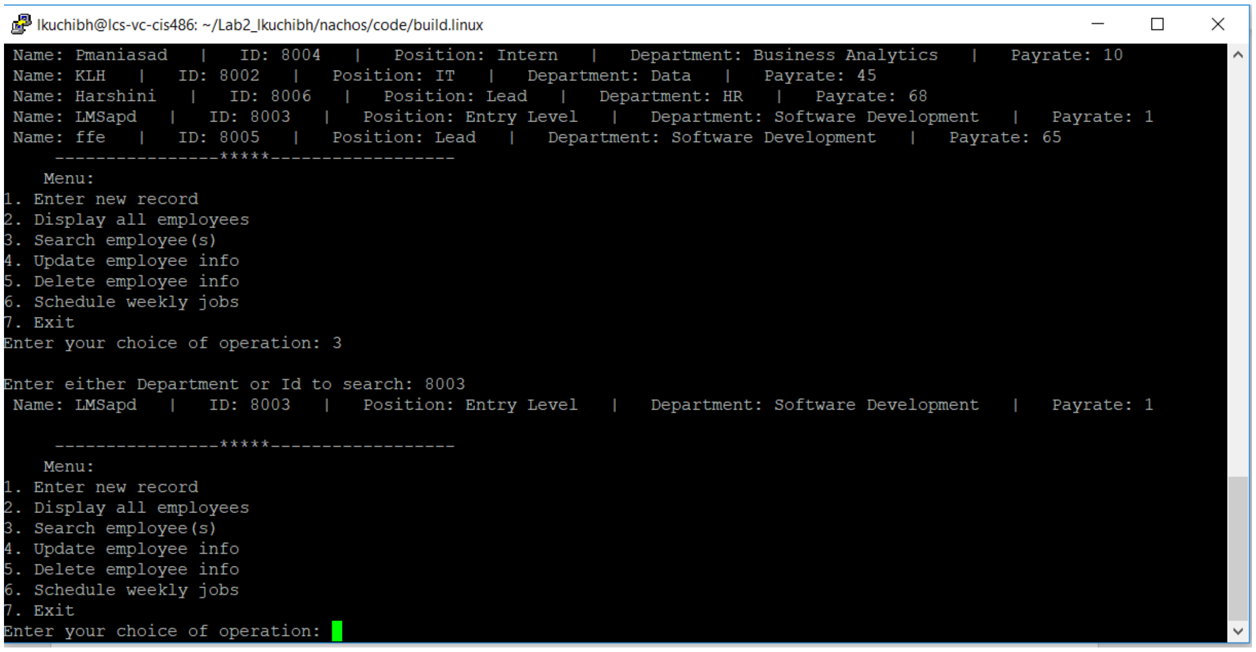


lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

```
Enter Name: Harshini

Choose from the mentioned Positions
1. Intern | 2. Entry Level | 3. Associate | 4. Senior | 5. Lead | 6.Manager
Enter Position(1/2/3/4/5/6): 1
1. Administrative | 2. HR | 3. Software Development | 4. Business Analytics

Enter Department(1/2/3/4): 3
Enter the payrate for the employee: 85
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: Harshini    |    ID: 8001    |    Position: Intern    |    Department: Software Development    |    Payrate: 85
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

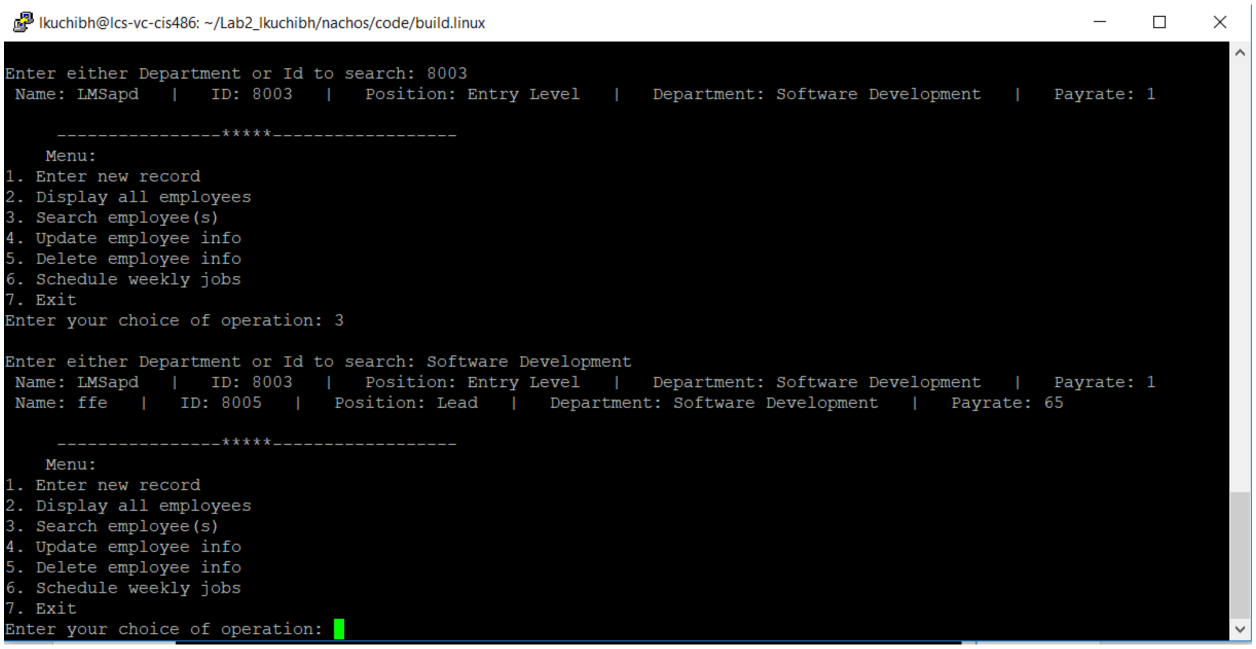
-Search Employees

Search using ID



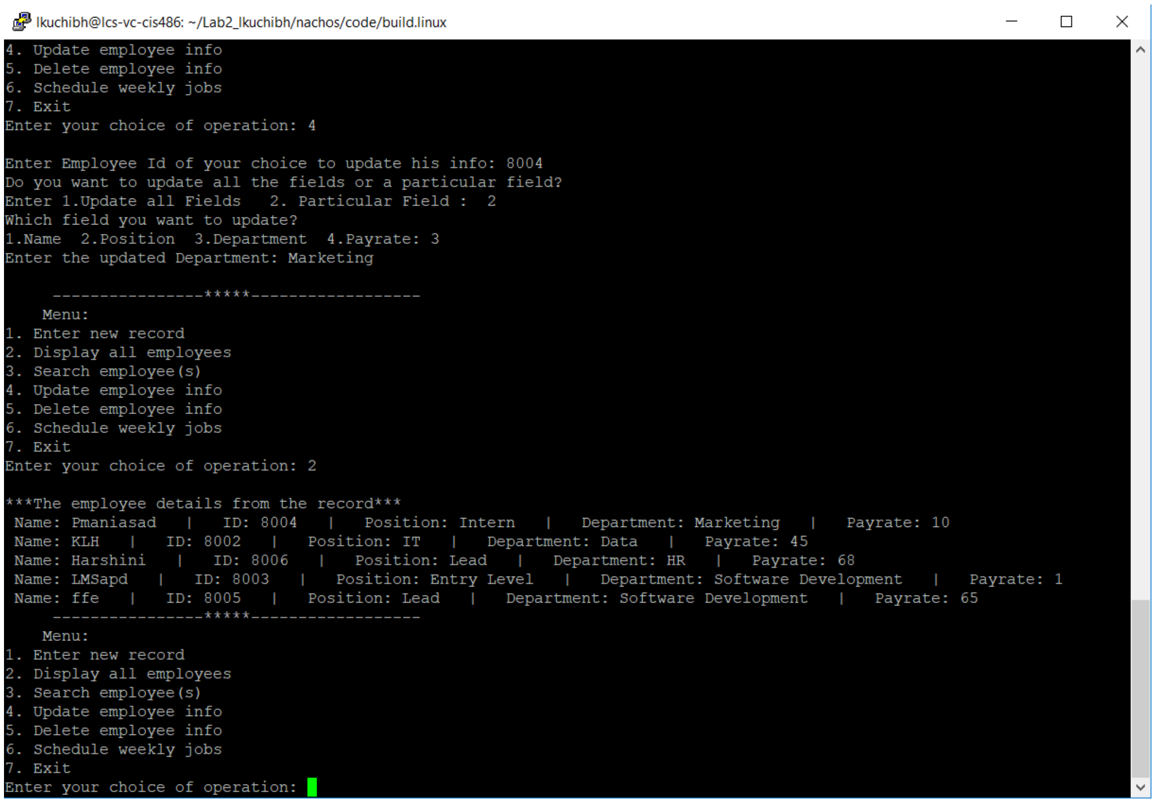
```
lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux
 Name: Pmaniasad   |   ID: 8004   |   Position: Intern   |   Department: Business Analytics   |   Payrate: 10
 Name: KLH   |   ID: 8002   |   Position: IT   |   Department: Data   |   Payrate: 45
 Name: Harshini   |   ID: 8006   |   Position: Lead   |   Department: HR   |   Payrate: 68
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1
 Name: ffe   |   ID: 8005   |   Position: Lead   |   Department: Software Development   |   Payrate: 65
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 3

Enter either Department or Id to search: 8003
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1

     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

Search using Department

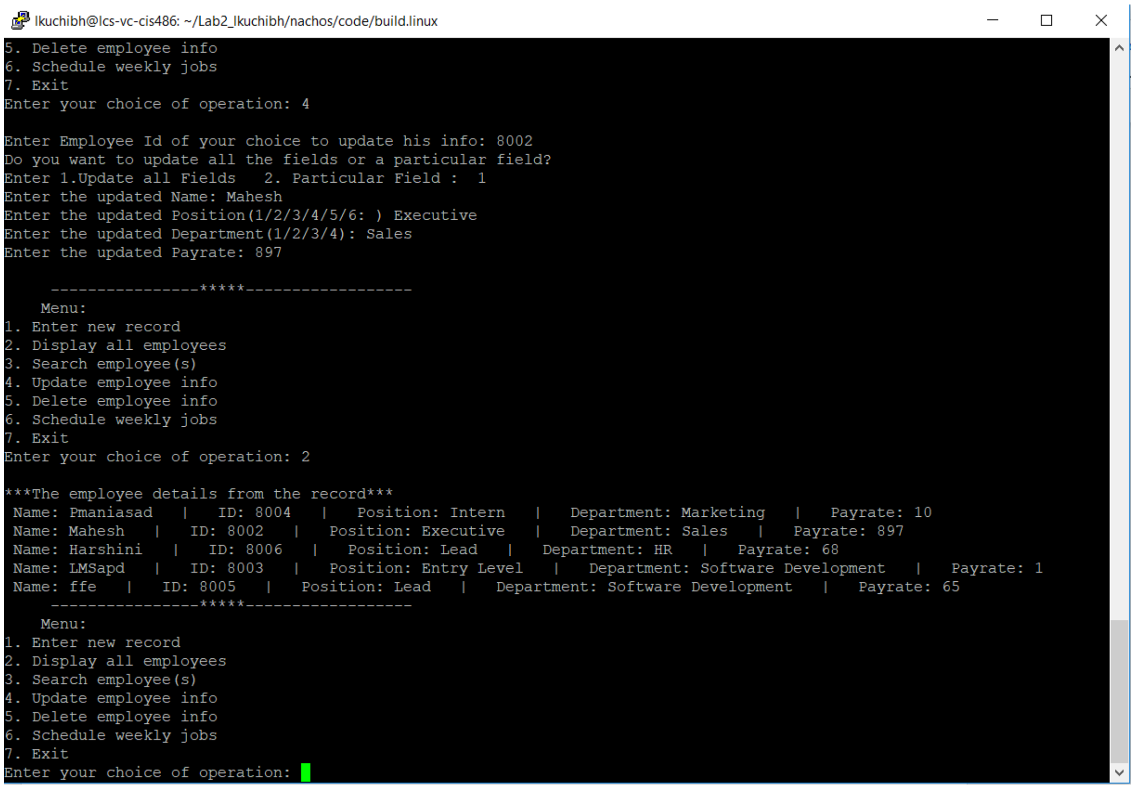


```
lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

Enter either Department or Id to search: 8003
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1

     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 3

Enter either Department or Id to search: Software Development
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1
 Name: ffe   |   ID: 8005   |   Position: Lead   |   Department: Software Development   |   Payrate: 65

     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

-Update Employee info

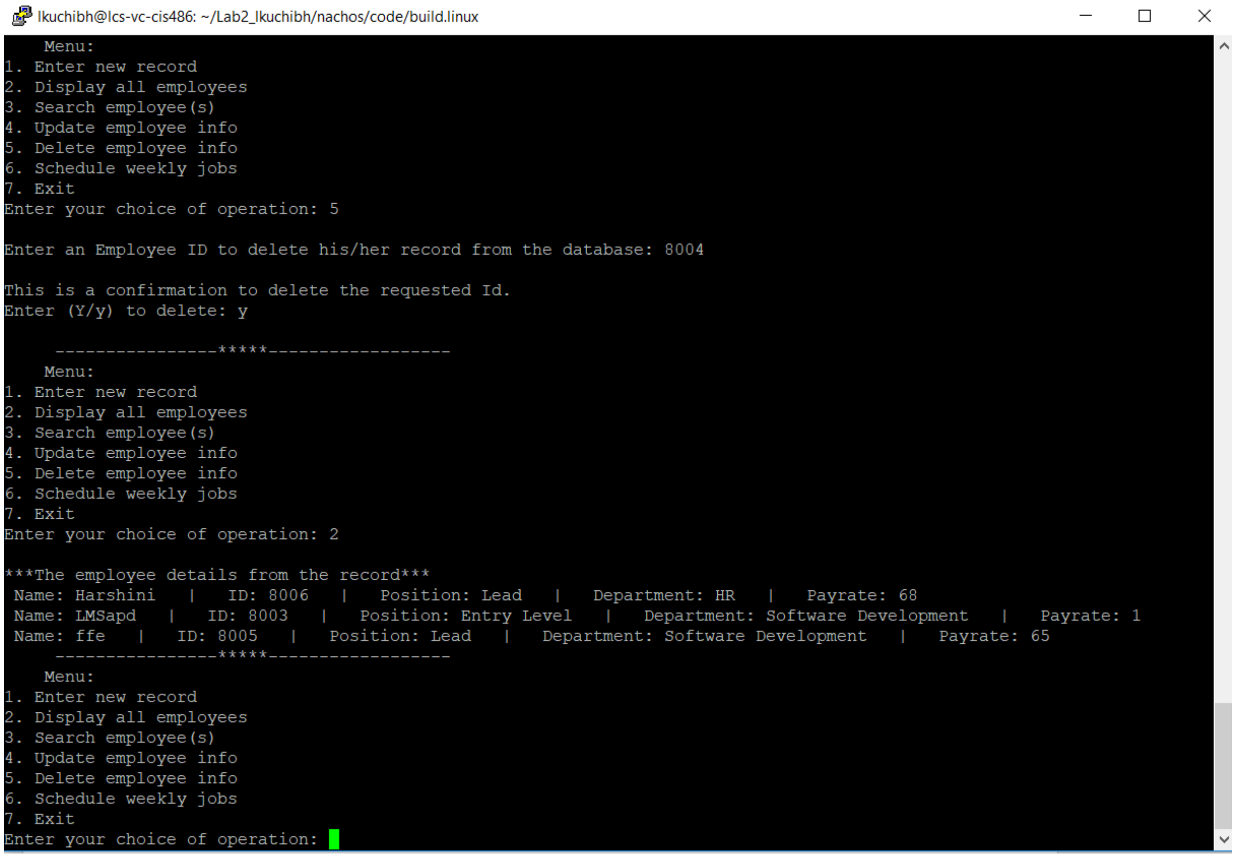

lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

```
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 4

Enter Employee Id of your choice to update his info: 8004
Do you want to update all the fields or a particular field?
Enter 1.Update all Fields   2. Particular Field :  2
Which field you want to update?
1.Name  2.Position  3.Department  4.Payrate: 3
Enter the updated Department: Marketing

     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: Pmaniasad   |   ID: 8004   |   Position: Intern   |   Department: Marketing   |   Payrate: 10
 Name: KLH   |   ID: 8002   |   Position: IT   |   Department: Data   |   Payrate: 45
 Name: Harshini   |   ID: 8006   |   Position: Lead   |   Department: HR   |   Payrate: 68
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1
 Name: ffe   |   ID: 8005   |   Position: Lead   |   Department: Software Development   |   Payrate: 65
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```


lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

```
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 4

Enter Employee Id of your choice to update his info: 8002
Do you want to update all the fields or a particular field?
Enter 1.Update all Fields   2. Particular Field :  1
Enter the updated Name: Mahesh
Enter the updated Position(1/2/3/4/5/6: ) Executive
Enter the updated Department(1/2/3/4): Sales
Enter the updated Payrate: 897

     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: Pmaniasad   |   ID: 8004   |   Position: Intern   |   Department: Marketing   |   Payrate: 10
 Name: Mahesh   |   ID: 8002   |   Position: Executive   |   Department: Sales   |   Payrate: 897
 Name: Harshini   |   ID: 8006   |   Position: Lead   |   Department: HR   |   Payrate: 68
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1
 Name: ffe   |   ID: 8005   |   Position: Lead   |   Department: Software Development   |   Payrate: 65
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

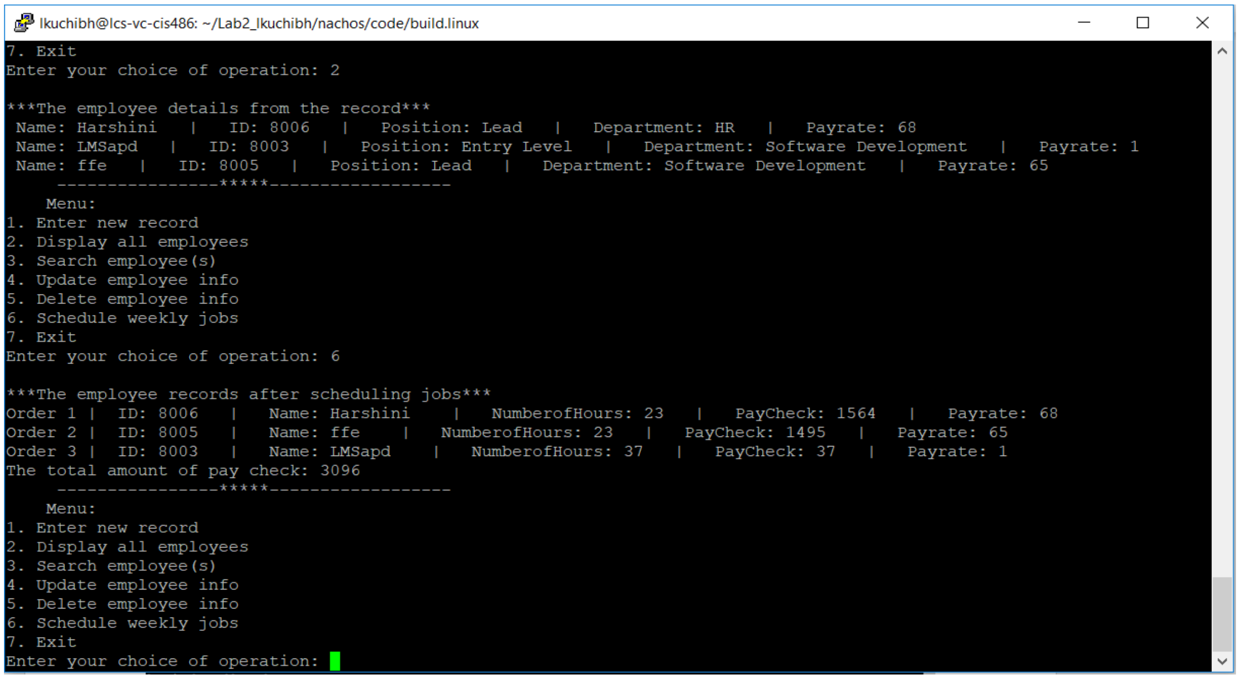
-Delete Employee Record



lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

```
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 5

Enter an Employee ID to delete his/her record from the database: 8004

This is a confirmation to delete the requested Id.
Enter (Y/y) to delete: y

     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: Harshini    |   ID: 8006    |   Position: Lead    |   Department: HR    |   Payrate: 68
 Name: LMSapd    |   ID: 8003    |   Position: Entry Level    |   Department: Software Development    |   Payrate: 1
 Name: ffe    |   ID: 8005    |   Position: Lead    |   Department: Software Development    |   Payrate: 65
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

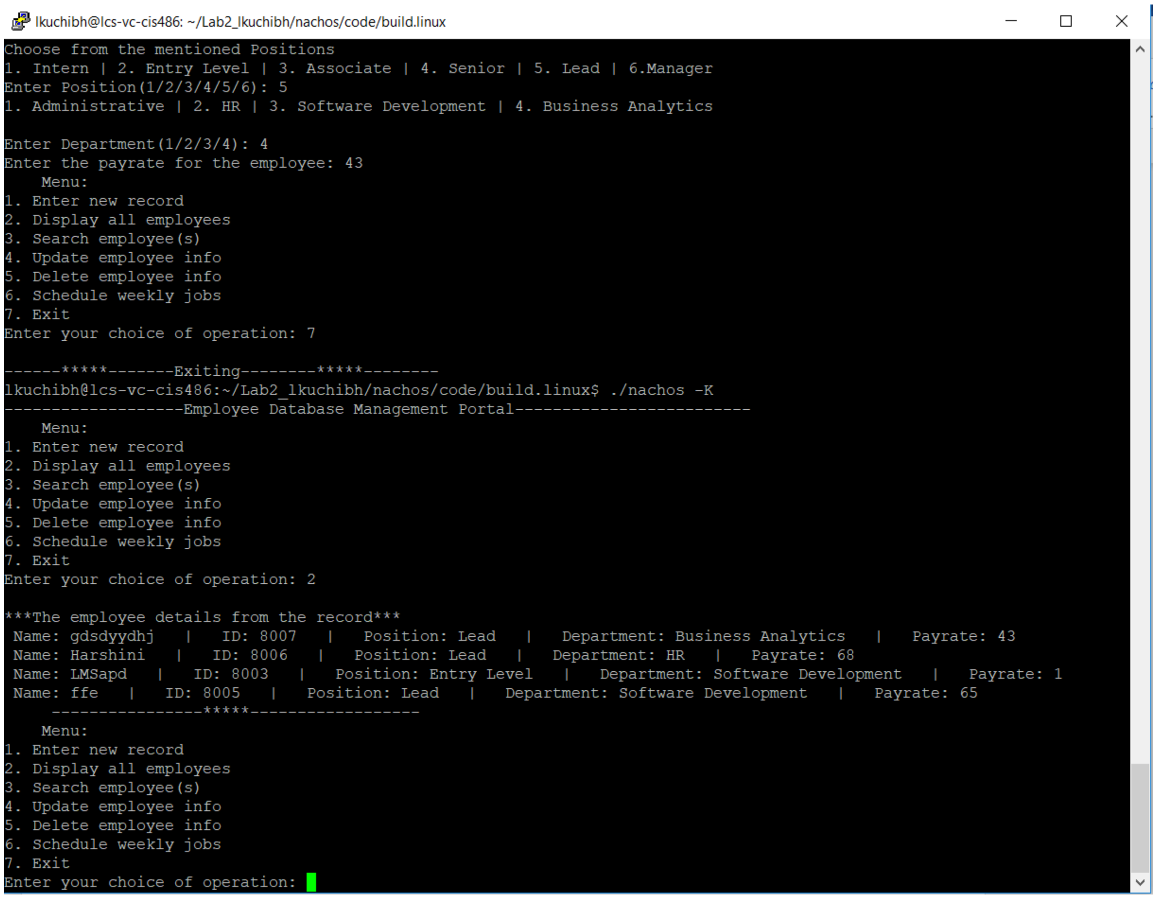
-Calculate paycheck of all the employees and the total paycheck after job is scheduled sorted with paycheck in descending order



lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

```
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: Harshini    |   ID: 8006    |   Position: Lead    |   Department: HR    |   Payrate: 68
 Name: LMSapd    |   ID: 8003    |   Position: Entry Level    |   Department: Software Development    |   Payrate: 1
 Name: ffe    |   ID: 8005    |   Position: Lead    |   Department: Software Development    |   Payrate: 65
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 6

***The employee records after scheduling jobs***
Order 1 |  ID: 8006    |    Name: Harshini    |    NumberofHours: 23    |    PayCheck: 1564    |    Payrate: 68
Order 2 |  ID: 8005    |    Name: ffe    |    NumberofHours: 23    |    PayCheck: 1495    |    Payrate: 65
Order 3 |  ID: 8003    |    Name: LMSapd    |    NumberofHours: 37    |    PayCheck: 37    |    Payrate: 1
The total amount of pay check: 3096
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation:
```

-Writeback to the employee.dat file before exiting the program


lkuchibh@lcs-vc-cis486: ~/Lab2_lkuchibh/nachos/code/build.linux

```
Choose from the mentioned Positions
1. Intern | 2. Entry Level | 3. Associate | 4. Senior | 5. Lead | 6.Manager
Enter Position(1/2/3/4/5/6): 5
1. Administrative | 2. HR | 3. Software Development | 4. Business Analytics

Enter Department(1/2/3/4): 4
Enter the payrate for the employee: 43
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 7

------*****-------Exiting--------*****--------
lkuchibh@lcs-vc-cis486:~/Lab2_lkuchibh/nachos/code/build.linux$ ./nachos -K
-------------------Employee Database Management Portal-------------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 2

***The employee details from the record***
 Name: gdsdyydhj   |   ID: 8007   |   Position: Lead   |   Department: Business Analytics   |   Payrate: 43
 Name: Harshini   |   ID: 8006   |   Position: Lead   |   Department: HR   |   Payrate: 68
 Name: LMSapd   |   ID: 8003   |   Position: Entry Level   |   Department: Software Development   |   Payrate: 1
 Name: ffe   |   ID: 8005   |   Position: Lead   |   Department: Software Development   |   Payrate: 65
     ----------------*****------------------
    Menu:
1. Enter new record
2. Display all employees
3. Search employee(s)
4. Update employee info
5. Delete employee info
6. Schedule weekly jobs
7. Exit
Enter your choice of operation: 
```