Standard Answer to Homework 8 for Linked Lists

```
/*************************************************************************/
/*                                                                       */
/*  Name:  Timothy Niesen                                                */
/*                                                                       */
/*  Assignment  Standard Version of Homework 8                           */
/*                                                                       */
/*  Date: 04/15/93                                                       */
/*                                                                       */
/*************************************************************************/

#include <stdio.h>

/* define local constants */
#define NAME_LEN     20
#define MAX_HRS   158.0
#define NORMAL_HRS 40.0
#define OT_SCALE    1.5

/*  Global structure declaration  */
struct employee
    {
    char name[NAME_LEN];    /*  the employee's name                    */
    long id_number;         /*  the employee's id #                    */
    float wage;             /*  the rate of pay for the employee       */
    float hours;            /*  the # of hours the employee worked     */
    float overtime;         /*  the # of overtime hours (if any) the   */
                            /*  employee worked                        */
    float gross;            /*  the employee's gross pay               */
    struct employee *next;  /*  pointer to the next node in linked list */
    };

/*************************************************************************/
/*                                                                       */
/*                        Function Get_Info                              */
/*                                                                       */
/*  Purpose:     This function will prompt the user to enter the         */
/*               employee's name, id number and wage.                    */
/*                                                                       */
/*  Parameters:  number_of_employees - number of employees               */
/*                                                                       */
/*  Returns:     this function will return a pointer to the first        */
/*               node in the linked list it created.                     */
/*  Side                                                                 */
/*  Effects:     this function creates and initializes a global          */
/*               linked list of employee structures.                     */
/*                                                                       */
/*************************************************************************/

struct employee *get_info (number_of_employees)

int number_of_employees;

    {

    struct employee *pointer;    /* pointer that points to current node */
    struct employee *head_ptr;   /* always points to the first node     */

    /*  make sure there is at least one employee  */

    if(number_of_employees > 0)
        {
        /*  create first node of linked list  */
        head_ptr = (struct employee *) malloc (sizeof(struct employee));

        /*  point temporary pointer to first node  */
        pointer = head_ptr;
```

```
        }
    else
        {
        /*  set head pointer to null  */
        head_ptr = NULL;

        /*  return to calling function  */
        return (head_ptr);
        }

    /*  loop until there are no employees left to input  */

    while(number_of_employees > 0)
        {
        /*  prompt user to enter the employee's name, id #, and wage  */
        /*  Note that name is an array name, and id and wage are long */
        /*  int and float, so they need the &.                        */
        printf ("\n");
        printf ("Please enter the employee's name:  ");
        scanf ("\n");
        gets (pointer->name);
        printf ("\nPlease enter %s's id number:  ", pointer->name);
        scanf ("%ld", &pointer->id_number);
        printf ("\nPlease enter %s's wage:  ", pointer->name);
        scanf ("%f", &pointer->wage);
        printf ("\n");

        /*  decrement the number of employees  */
        number_of_employees = number_of_employees - 1;

        /*  test to see if another node should be created  */
        if(number_of_employees > 0)
            {
            /*  create another node  */
            pointer->next=(struct employee *) malloc (sizeof(struct employee));

            /*  move temporary pointer to new node  */
            pointer = pointer->next;
            }
        }  /*  end of while loop  */

    /*  set the last pointer to null  */
    pointer->next = NULL;

    /*  return head pointer  */
    return (head_ptr);
    }  /*  end of function get_info  */


/*********************************************************************/
/*                                                                   */
/*                        Function Prompt_User                       */
/*                                                                   */
/*  Purpose:     This function will prompt the user with the employee    */
/*               name and id number for the hours that employee worked.  */
/*                                                                   */
/*  Parameters:  head_ptr - pointer to the head of the linked list       */
/*                          of employee information                  */
/*               number_of_employees  - number of employees              */
/*                                                                   */
/*  Side                                                             */
/*  Effects:     the global linked list of employee information has      */
/*               the hours of each employee updated.                 */
/*                                                                   */
/*********************************************************************/

void  prompt_user (head_ptr,number_of_employees)
```

```
struct employee *head_ptr;
int number_of_employees;

    {
    int i;                        /* counter         */
    struct employee *pointer;     /* working pointer */

    /*  set temporary pointer to the head of the linked list  */
    pointer = head_ptr;

    /*  loop for each employee  */
    for(i = 0; i < number_of_employees; ++i)
        {
        /*  prompt user for the hours worked for each employee  */

        printf ("\nPlease enter the hours worked for %s employee # %.6ld:  ",
          pointer->name, pointer->id_number);
        scanf ("%f", &pointer->hours);
        while(pointer->hours > MAX_HRS || pointer->hours < 0.0)
            {
            /*  Test for a valid number of hours worked  */
            if(pointer->hours > MAX_HRS)
                printf ("Impossible to work that many hours in one week\n");
            else
                /*  entered a negative number  */
                printf ("Can not work a negative amount of hours\n");

            /*  prompt user to enter the hours again until a valid
            number of hours is reached  */

            printf ("\nPlease enter the hours worked for %s employee # %.6ld:  ",
            pointer->name, pointer->id_number);
            scanf ("%f", &pointer->hours);
            }  /*  end of while loop  */

        /*  move the temporary pointer to the next node in the linked list  */
        pointer = pointer->next;

        }  /*  end of for loop */

    /*  return to calling function  */
    return;

    }  /*  end of function  */


/*************************************************************************/
/*                                                                       */
/*                        Function Over_Hours                            */
/*                                                                       */
/*  Purpose:     This function will calculate the over time hours for     */
/*               for each employee if necessary.                          */
/*                                                                       */
/*  Parameters:  emp_hours - the # of hours the employee worked           */
/*                                                                       */
/*  Returns:     the number of overtime hours worked by the employee      */
/*                                                                       */
/*************************************************************************/

float over_hours (emp_hours)

float emp_hours;

    {

    /*  test to see if any overtime was worked  */
    if(emp_hours > NORMAL_HRS)
        return (emp_hours - NORMAL_HRS);
```

```
    else
        /*  no overtime was worked  */
        return (0.0);
    }  /* end of function over_hours  */



/*************************************************************************/
/*                                                                       */
/*                        Function Over_Pay                              */
/*                                                                       */
/*  Purpose:     This function will calculate the over time pay for      */
/*               each employee if necessary.                             */
/*                                                                       */
/*  Parameters:  wage   - rate of pay for the employee                   */
/*               ot_hrs - number of overtime hours the employee worked   */
/*                                                                       */
/*  Returns:     overtime pay for the employee                           */
/*                                                                       */
/*************************************************************************/

float over_pay (wage, ot_hrs)

float wage, ot_hrs;

    {
    float ot_rate,    /* ther rate the overtime is based on, such as */
                      /* time and a half                             */
          ot_pay;     /* The amount of pay that is from overtime     */

    /*  calculate the overtime pay  */
    ot_pay = ot_hrs * (wage * OT_SCALE);

    /*  return the overtime pay to the calling function  */
    return (ot_pay);

    }  /*  end of function over_pay  */

/*************************************************************************/
/*                                                                       */
/*                        Function Calc_Gr_Pay                           */
/*                                                                       */
/*  Purpose:     This function will calculate the gross pay for an       */
/*               employee.  It will add to the gross pay any overtime    */
/*               pay if necessary.                                       */
/*                                                                       */
/*  Parameters:  wage      - rate of pay for the employee                */
/*               hours     - # of hours the employee worked              */
/*               ot_pay    - overtime pay for the employee               */
/*                                                                       */
/*  Returns:     gross pay (including any overtime pay) for the employee */
/*                                                                       */
/*************************************************************************/

float calc_gr_pay (wage, hours, ot_pay)

float wage, hours, ot_pay;

    {
    float gross_pay;    /* gross pay = wage * hours + overtime pay */

    /*  test to see if any overtime was worked  */
    if(hours > NORMAL_HRS)

        /*  calculate gross pay */
        gross_pay = NORMAL_HRS * wage + ot_pay;

    else /* no overtime worked  */
```

```
        /*  calculate gross pay  */
        gross_pay = wage * hours;

    /*  return the value of gross_pay to call function  */
    return (gross_pay);

    }  /*  end of function calc_gr_pay  */


/*********************************************************************/
/*                                                                   */
/*                        Function Print_Results                     */
/*                                                                   */
/*  Purpose:     This function will print the results in the form of  */
/*               a table.                                            */
/*                                                                   */
/*  Parameters:  head_ptr - pointer to the head linked list of       */
/*                          employee information                     */
/*               number_of_employees  - number of employees          */
/*                                                                   */
/*********************************************************************/

void print_results (head_ptr, number_of_employees)

struct employee *head_ptr;
int number_of_employees;

    {

    int i;
    struct employee *pointer;

    /*  print column headings  */
    printf("\n\n\n");
    printf("-------------------------------------------------------------      -----------------");
    printf("\n");
    printf("     NAME          CLOCK #       WAGE      HOURS          OT      GROSS \n");
    printf("-------------------------------------------------------------      -----------------");
    printf("\n\n");

    /*  set temporary pointer to first node in linked list  */
    pointer = head_ptr;

    /*  print results  */
    for(i = 0; i < number_of_employees; ++i)
        {
        printf ("%-15s    %.6ld     %7.2f    %7.1f    %7.1f     %8.2f \n\n\n",
                  pointer->name,pointer->id_number,pointer->wage,
                  pointer->hours,pointer->overtime,pointer->gross);

        /*  move temporary pointer to next node in linked list  */
        pointer = pointer->next;

        }  /*  end of for loop  */

    /*  return to calling function  */
    return;

    }  /*  end of function print_results  */


/*********************************************************************/
/*                                                                   */
/*                        Function Main                              */
/*                                                                   */
/*  Purpose:     This function will calculate the gross pay for five  */
/*               employees.  It will call the necessary functions to  */
/*               accomplish this task.                               */
```

```
/*                                                                      */
/*  Functions                                                           */
/*     Called:   get_info      - will prompt the user to enter the      */
/*                               employee's name, id #, and wage        */
/*               prompt_user   - will prompt user to enter the hours    */
/*                               worked for each employee.              */
/*               over_hours    - will calculate the overtime hours, if  */
/*                               any, for each employee.                */
/*               over_pay      - will calculate the overtime pay,if any,*/
/*                               for each employee.                     */
/*               calc_gr_pay   - will calculate the gross pay for each  */
/*                               employee.                              */
/*               print_results - will print the final results in a table*/
/*                                                                      */
/************************************************************************/

main ()
    {

    /*  variable declaration    */
    struct employee *emp_ptr;     /* always pointer to the first node */
    struct employee *temp_ptr;    /* points to the current node       */
    float ot_pay;                 /* overtime pay                     */
    int i,                        /* loop counter                     */
        num_emp;                  /* total number of employees        */

    /*  prompt user to enter the number of employee  */
    printf ("Please enter the number of employees:  ");
    scanf ("%d", &num_emp);

    /*  call function to get the employee information  */
    emp_ptr = get_info (num_emp);

    /*  test to see if there are any employees  */
    if(num_emp == 0)

        /* print message to operator  */
        printf ("no results due to no employees\n");

    else
        {
        /* call local function to prompt the user to enter the employee hours */
        prompt_user (emp_ptr, num_emp);

        /*  set temporary pointer to first node in linked list  */
        temp_ptr = emp_ptr;

        /*  loop for each employee  */
        for(i=0; i<num_emp; ++i)
            {
            /*  initialize the overtime pay  */
            ot_pay = 0.0;

            /*  test to see if the employee worked any overtime  */
            if(temp_ptr->hours > NORMAL_HRS)
                {
                /*  call local function to calculate the overtime hours  */
                temp_ptr->overtime = over_hours (temp_ptr->hours);

                /*  call local function to calculate the overtime pay  */
                ot_pay = over_pay (temp_ptr->wage, temp_ptr->overtime);
                }
            else
                {
                temp_ptr->overtime = 0.0;
                }  /*  end of if statement  */

            /*  call local funcion to calculate the gross pay  */
```

```
        temp_ptr->gross = calc_gr_pay (temp_ptr->wage,temp_ptr->hours,ot_pay);
        /*  move temporary pointer to next node in the linked list  */
        temp_ptr = temp_ptr->next;

        }  /*  end of for loop  */

    /*  call local function to print the results  */
    print_results (emp_ptr, num_emp);

    }  /*  end of if any employees  */

}  /* end main */
```