

Bangladesh University of Business and Technology (BUBT)

**A Report on Mini Project**

**“Employee Record System”**

**Submitted By: Submitted To:**

1. Rafiqul Islam Rafi (138)

2. Mohammad Yeasin Arafat (101)

3. Syeda Farhana Aziz (102)

4. Fazley Rabbi Ripon (099)

5. Marzia Akter Mita (112)

Ahmed Shoeb Al Hasan

Asst. Professor

Dept. of CSE

Bangladesh University of Business & Technology (BUBT)

## 

## ACKNOWLEGDEMENT

We would like to show our sincere gratitude to the department of CSE for providing us a chance to develop a project under C.

We owe our deepest gratitude to Sir Ahmed Shoeb Al Hasan

, our teacher of Software Development-I who gave us the concepts of programming in C and also suggested various technical ideas useful for our project. We further cannot withstand without thanking our dear friends, especially team members and seniors for their advice, encouragement, support and help.

We would like to deeply thank the various people who provided us with the useful and helpful assistance for the initiation of this project.

And finally, we would like to thank all of the persons, who directly or indirectly helped us in the project.

Date: 14-02-2016

Project Members:

1. Rafiqul Islam Rafi (138)

2. Mohammad Yeasin Arafat (101)

3. Syeda Farhana Aziz (102)

4. Fazley Rabbi Ripon (099)

5. Marzia Akter Mita (112)

# Objectives

The project is mainly based on following objectives:

* To create a project using C-programming and its features.
* To implement features like control statement, structures and file handlings.
* To be familiar with resource reusability by making user defined function.
* To make the program easy while running it.
* To concise the memory of program as far as possible.
* To get an idea about making a simple project using C.
* To be able to solve problems by Compiling and Debugging.

# Introduction

The mini project ‘Employee Record System’ is the sample project for managing employee information of a company. The project aims at developing employee record system using the C language that enables an organization to maintain it.

The project demonstrates the creation of a user interface of a system, without the use of C Graphics library.

The application uses basic C function to generate menus, show message boxes and print text on the screen.

To display customized text with colors and fonts according to application requirements, functions have been created in the application, which fetch the exact video memory addresses of a target location, to write text at a particular location.

The application also implements the concept of structures to define the library items. It also effectively applies the various C concepts such as file operations, looping and branching constructs and string manipulation functions.

# Introduction to c

**C** was evolved from **ALGOL**, **BCPL** and **B** by Dennis Ritchie at the bell laboratories in 1972. **C** uses many concepts from these languages and added the concept of data types and other powerful features.

**C** is a powerful, flexible, portable and elegantly structured programming language. The increasing popularity of c is probably due to its many desirable quantities. It is a robust language whose rich set of built in functions and operators can be used to write any complex of a high level language and therefore it is well suited for writing both system software and business packages.

**C** is highly portable. This means that c programs written for one computer can be run on another with little or no modification. Portability is important if we plan to use a new computer with a different operating system.

**C** language is well suited for structured programming, thus requiring the user to think of a problem in terms of function modules or blocks. A proper collection of these modules would make a complete program. This modular structure makes program debugging, testing and maintenance easier.

Another important feature of c is its ability to extend itself. A c program is basically a collection of functions that are supported but the c library. We can continuously ad our own functions to **C** library. With the availability of a large number of functions the programming task becomes simple.

To assure that the **C** language remains standard, in 1983, American national standards institute (**ANSI**) appointed a technical committee to define a standard for **C**. the committee approved but the international standards organization (**IDO**) in 1990 which is now known as **ANSI C** .it was then approved but the international standards organization (**ISO**) in 1990. This version of c is also referred to as **C89**.

## Control statement

C language possesses such decision making capabilities by supporting the follow statements:

* If statement
* Switch statement
* Conditional operator statement

These statements are popularly known as decision-making statements. Since these statements ‘control’ the flow execution, they are known as control statements.

1. **control statement**

The if-statement is a powerful decision making statement and is used to control the flow of execution of statements. It is basically a two-way decision statement and is used in conjunction with an expression. It takes the following form:

If(test expression)

It allows the computer to evaluate the expression first and then, depending on whether the value expression (relation or condition) is ‘true’ (or non-zero) of ‘false’ (zero),it transfers the control to a particular statement. This point of program has two paths to follow, one for the true condition and the other for the false condition as shown below:

Entry

Test

Expression

?

false

true

The if statement may be implemented in different forms depending on the complexity of conditions to be tested.the different forms are:

* Simple if statement
* If……else statement
* Nested if………else statement
* Else if ladder.

**2**. Switch statement:-

C has a built-in multi way decision statement known as a switch. The switch statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated with that case is executed. The general form of the switch statement is as shown below:

switch (expression)

{

case value-1:

block-1

break;

case value-2:

block-2

break;

………..

……….

default:

default-block

break;

}

statement-x;

The expression is an integer expression or characters.value-1,value-2…..are constants or constant expressions (evaluable to an integral constant) and are known as case labels. Each of these values should be unique within a switch statement. block-1,block-2…..are

Statement lists and may contain zero or more statements. There is no need to put braces around these blocks. Note that case labels end with a colon(:).

**3.** Conditional operator statement**:-**

The c language has an unusual operator, useful for making two-way decisions. This operator is a combination of? And takes three operands. This operator is popularly known as the conditional operators. The general form of use of the conditional operator is as follows:

Conditional expression? expression1:expression2

The conditional expression is evaluated first. if the result is nonzero,expression1 is evaluated and is returned as the value of the conditional expression.Otherwise,expression2 is evaluated and its value is returned.

## Function

Most languages allow you to create functions of some sort. Functions are used to break up large programs into named sections. You have already been using a function which is the main function. Functions are often used when the same piece of code has to run multiple times.

In this case you can put this piece of code in a function and give that function a name. When the piece of code is required you just have to call the function by its name. (So you only have to type the piece of code once).

In the example below we declare a function with the name MyPrint. The only thing that this function does is to print the sentence: Printing from a function. If we want to use the function we just have to call MyPrint() and the printf statement will be executed. (Don’t forget to put the round brackets behind the function name when you call it or declare it).

Take a look at the example:

#include<stdio.h>

void MyPrint()

{

printf("Printing from a function.\n");

}

int main()

{

MyPrint();

return 0;

}

**Parameters and return**

Functions can accept parameters and can return a result. (C functions can accept an unlimited number of parameters).

here the functions are declared in your program does not matter, as long as a functions name is known to the compiler before it is called. In other words: when there are two functions, i.e. functions A and B, and B must call A, than A has to be declared in front of B.

Let’s take a look at an example where a result is returned:

#include<stdio.h>

int Add(int output1,int output2 )

{

printf("%d", output1);

printf("%d", output2);

return output1 + output2;

}

int main()

{

int answer, input1, input2;

scanf("%d", &input1);

scanf("%d", &input2);

anwser = Add(input1,input2);

printf(" answer = %d\n", answer);

return 0;

}

The main() function starts with the declaration of three integers. Then the user can input two whole numbers. These numbers are used as input of function Add(). Input1 is stored in output1 and output2 is stored in Second. The function Add() prints the two numbers onto the screen and will return the result of output1 + output2. The return value is stored in the integer answer. The number stored in answer is then printed onto the screen.

**Void**

If you don’t want to return a result from a function, you can use void return type instead of the int.  
So let’s take a look at an example of a function that will not return an integer:

void our\_site()

{

printf("www");

printf(".NextDawn");

printf(".nl");

}

**Note:** As you can see there is not an int before our\_site() and there is not a return 0; in the function.

The function can be called by the following statement: our\_site();

**Global and local variables**

A local variable is a variable that is declared inside a function. A global variable is a variable that is declared outside **all** functions. A local variable can only be used in the function where it is declared. A global variable can be used in all functions.

See the following example:

#include<stdio.h>

// Global variables

int A;

int B;

int Add()

{

return A + B;

}

int main()

{

int answer; // Local variable

A = 5;

B = 7;

answer = Add();

printf("%d\n",answer);

return 0;

}

As you can see two global variables are declared, A and B. These variables can be used in main() and Add().  
The local variable answer can only be used in main().

That’s all for this tutorial

More about function

In this C programming language tutorial we will talk some more about functions. We will take a look at command-line parameters and function prototypes.

**Command-line parameters**

In some cases you want to give a parameter at the start of a program.  
For example:

# myprogram -i

The program myprogram will start and something extra will be done because of the command-line parameter -i (What it will do is up to you, this is just an example).

Now let’s make a program that will print the words that are typed behind the program at execution time. (Compile the program with the name myprogram).  
Here is the example:

#include<stdio.h>

int main(int argc, char \*argv[])

{ int x;

printf("%d\n",argc);

for (x=0; x < argc; x++)

printf("%s\n",argv[x]);

return 0;

}

After compiling the program “myprogram” start it as follows:

# myprogram aa bb cc dd

In this code, the main program accepts two parameters, argc and argv. The argv parameter is an array of pointers to a string that contains the parameters entered when the program was invoked at the UNIX command line. (Pointers will be explained in a later tutorial, for now it is enough to know that it points to an address in memory where the parameters are stored). The argc integer contains a count of the number of parameters. (In this case four).

First the program will print the number of parameters given when the program was invoked (stored in argc). This number will be used in a “for loop”. In this case it will print four times.

The second printf statement will print the parameters given when the program was invoked, one by one. **Try it!** Command-line parameters can be used for many things.

**Function prototypes**

A function prototype declares the function name, its parameters, and its return type to the rest of the program. This is done before a function is declared. (In most cases at the beginning of a program).  
To understand why function prototypes are useful, try the following program:

#include<stdio.h>

void main()

{

printf("%d\n",Add(3)); /\* <- There is the error ! \*/

}

int Add(int a, int b)

{

return a + b;

}

**Note:** just one parameter when the function is called. (Not a typo).

The example above will be compiled on many of the compilers. They just give some warning, because the function Add() needs two parameters as input, not one. The result of this program cannot be foreseen. It works because many C compilers do not check for parameters matching either in type or count. The result is that you will spend a lot of time debugging programs, because you made a mistake of passing on to many or too few parameters to a function.

To solve this problem you can make use of function prototypes. If you use prototypes, C checks the types and count of the parameter list.

Try the next example:

#include<stdio.h>

int Add (int,int); /\* function prototype for Add \*/

void main()

{

printf("%d\n",add(3)); /\* <- There is the error ! \*/

}

int Add(int a, int b)

{

return a + b;

}

When you try to compile this program, the compiler will flag an error on the printf statement.

So in the future use function prototypes for every function you declare. It will save you a lot of time!

## Arrays and pointers

### Array

An array can be defined as a group of homogeneous elements sharing a common name.

In many of the programming situations, we may require the processing of data items that have common characteristics. now in such case, it would be easier if we place these data items into one single variable called array which is capable of storting number of data, sharing common name.

The individual data items can be characters, integers, and floating-point numbers and so on they must all, however be of the same type and the same storage class. The individual data items in an array are called array elements.

Each array element is referred to by specifying the array name followed by one or more subscript enclosed in square brackets. Each subscript or index must be expressed as non-negative integer.

Thus, we represent array containing elements as:

x[n]

Where, x is array name

N is subscript &

x[n] has its array element as x[0],x[1],x[2],………x[n-1].

The value of each subscript can be expressed as an integer constant,integer variable or a more complex integer expression. The number of subscript determines the dimensionality

of the array.

### Pointers

A pointer is a derived data type in C. It is built from one of the fundamental data types available in C. Pointers contain memory addresses as their values. Since these memory addresses are the locations in the computer memory where program instructions and data are stored, pointers can be used to access and manipulate data stored in the memory.

Pointers are undoubtedly one of the most distinct and exciting features of c language. It

has added power and flexibility to the language. Although they appear little confusing and difficult to understand for a beginner, they are a powerful tool and handy to use once they are mastered.

**Pointer Declaration:-**

Pointer variable like other variables must be declared before they may be used in a C Program. The interpretation of pointer declaration differs, however from the interpretation of other variable declarations. When a pointer variable is declared, the variable name must be preceded by an asterisk (\*). This identifies the fact that the variable is a pointer.

A pointer declaration may be written in general terms as

*Data-type \*ptvar;*

Where ptvar is the name of the pointer, and data-type refers to the data type of the pointer’s object. Remember that an asterisk must precede ptvar.

**Passing pointer to a function:-**

Pointer are often passed to a function as arguments. This allows data items within the calling portion of the program to be accessed by the program in altered form. We refer to this use of pointers as passing arguments by reference, in contrast to passing arguments by value.

**Arrays of pointers:-**

One important use of pointers is in handling of a table of strings. Consider the following array of strings:

char name [3][25];

This says that the name is a table containing three names, each with a maximum length of 25 characters (including null character). The total storage requirements for the name table are 75 bytes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

We know that rarely the individual strings will be of equal lengths. Therefore, instead of making each row a fixed number of characters, we can make it a pointer to a string of varying length. For example,

char \*name[3]={“New Zealand”,”Australia”,”Nepal”};

declares name to be an array of three pointers to characters, each pointer pointing to a particular name as:

name[0] New Zealand

name[1] Australia

name[2] Nepal

This declaration allocates only 28 bytes, sufficient to hold all the characters as shown

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **e** | **w** |  | **Z** | **e** | **a** | **l** | **a** | **n** | **d** | **\0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **u** | **s** | **t** | **r** | **a** | **l** | **i** | **a** | **\0** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N** | **e** | **p** | **a** | **l** | **\0** |

The following statement would print out all the three names:

for ( i=0; i<=2; i++)

printf(“%s\n”, name[i] );

to access he jth  character in the ith name, we may write as

\*(name[i]+j);

The character arrays with the rows of varying length are called ‘ragged arrays’ and are better handled by pointers.

# **Structure**

C is provided with a constructed data type known as structures, a mechanism for packing data of different types. A structure is a convenient tool for handling a group of logically related data items. For example, it can be used to represent a set of attributes, such as student \_ name, roll\_ number and marks. The concept of a structure is analogous to that of a ‘record’ in many other languages.

Structures help to organize complex data in a more meaningful way. It is a powerful concept that we may often need to use in our program design. This chapter is devoted to the study of structures and their applications in program development.

In general, the syntax for structure definition is:

struct struct\_name

{

data \_ type mem1;

data \_ type mem2;

…………………………

………………………...

data \_type memn;

};

The structure definition starts with keyword struct followed by an identifier or tag name. The tag name is structure name and can e used for instantiating structure variable. In above definition, struct \_name is referred to as structure name or structure tag name; and mem1, mem2, memn are known as structure members or elements or fields. These members are enclosed within braces and terminated by semicolon.

After the structure has been specified, the structure variable can be declared as standard data type:

struct struct\_name var1, var2,………,varn;

The structure definition serves as a template for user defined data type. It does not reserve memory unless a variable of structure data type is declared.

**Accessing Member of structures**

In C programming language, the members of a structure are processed individually as separate entities. We make use of period or dot “.” Operator to access the individual members of structure.The syntax for accessing member of a structure variable is follows:

struct \_variable. member

Where struct \_ variable refers to the name of a structure variable, and member refers to the name of member within the structure. The dot (.) is an operator that separates the variable name from the member name. We noticed that the dit operator must have precedence among all operators and has left to right associatively. Consider the following statement:

struct employee e1;

Now, each member of the structure variable e1 can be accessed using the dot operator as follows:

e1.emp\_id

The employee’s employee IDnumer is accessed;

e1.name

The employee’s name is accessed. e1.salaryThe employee’s salary is accessed.

# File Handling

The console function like printf() and scanf() have been used for input/output .This scheme is adequate if the volume of data involved in not so large or it is not necessary to store the information for further use. However, many applications may require a large amount of data to be read, processed, and also saved for later use. Such information is stored on the auxiliary memory device in the form of data file.And a file is a collection of bytes that is given a name. In most computer systems, files are used as a unit of storage primarily on floppy-disk or fixed-disk data storage system (or they can be CDs or other storage device). Thus data files allow us to store information permanently, and to access and alter that information whenever necessary.

The file handling function available in standard library in order to implement I/O midel is classified as follows:

a) File access

b) Operation input/output

c) Formatted input/output

d) Character input/output

e) Direct input/output

f) File positioning

g) Error handling

The file access included the function like fopen() to open a file, fclose() to close a file , fflush () to flush out the buffer associated with a file, and freopen() to change the file associated with a stream. Also setvbuf() and setbuf() functions are use to allow the users explicitly control the file buffering strategy.

The operation on file includes like remove() to remove a file, remname() to rename a file ,tempfile() to create a temporary binary file and tmpnam() to generate a unique filename.

Formatted input/output group includes the n functions fscanf(), scanf() and sscanf() to read formatted data. Similarly fprintf() ,printf(), sprint(), vfprintf(), vprintf() and vsprintf() to write formatted data.

The character input/output group includes the functions fgetc() ,getc() and getchar() to read a character from an input stream and functions ungetc() to push back a character to an input stream. The functions fgets() and gets() are to read strings and the output functions fputc(),putc(), putchar(), fputs() and puts() are also included in this group.

The direct input/output group includes functions fread() to read and fwrite() to write a certain number of data items specified size.

File positioning group includes functions fread() to read and fwrite() to write a certain number if data items specified size.

File positioning group includes functions to set the file positon to some specified value to allow access to a specific portion of the seek(),interrogate the current file position ftell(),and reset the file position to the beginning of the file rewind().

Error handling group include functions to test whether EOF returned by a function indicates an end-of-file or an error (feof and ferror), clear end-of-file and indicators clearer, and map the error number errno to an error message perror.

**File Accessing:-**

To work with file using any file handling library functions, C requires a pointer to structure FILE defined in <stdio.h>.The declaration of file pointer is done as follows:

FILE \*fptr;

This declaration creates a variable fptr, which is pointed to the FILE structure. The pointer to the FILE structure is also called file pointer. The FILE structure describes the current state of a file such as file status flag, file descriptor, file buffer etc.

There are also predefined file pointers such stdin, stdout, and stderr which refers to standard input (normally keyboard), standard output (normally monitor) and standard error (connected to screen for error handling).

The file access functions provide the facilities to open and close a file, flush out the file buffer, change the file content and control the file buffering strategy.

Before working with a file it should be opened first. To open a file and associate it with a stream, we use fopen().Its prototype is shown here:

FILE \*fopen(char \*fname,char \*mode);

The fopen() function, like all the file-system functions ,uses the header <stdio.h> .The name of the file including the path is specified by fname and the mode specifies how file is accessed. Both of these parameters are string. The string specifying the mode is shown in following table.

|  |  |  |
| --- | --- | --- |
| **S.N** | **Mode** | **Meaning** |
| 1 | "r" | Open a text file for reading |
| 2 | "w" | Create a text file for writing |
| 3 | "a" | Append to a text file |
| 4 | "rb" | Open a binary file for reading |
| 5 | "wb" | Open a binary file for writing |
| 6 | "ab" | Append to binary file |
| 7 | "r+" | Open a text file for read/write |
| 8 | "w+" | Create a text file for read/write |
| 9 | "a+" | Append or creat a text file for read/write |
| 10 | "r+b" | Open a binary file for read/write |
| 11 | "w+b" | Create a binary file for read/write |
| 12 | "a+b" | Append a binary file for read/write |

# Algorithm

**Main function**

**steps**

1. Start with a System is password protected screen
2. Get password from user
3. Check password. Is password correct

then goto step 4

if password wrong then print Sorry! You entered wrong password try again and goto step 2

1. Display mainmenu as below

* 1.add new records
* 2.view records
* 3.search employee
* 4.modify or edit records
* 5.delete or remove records
* 6.calculate salary
* 7.calculate bonus
* 8.about
* 0.exit

1. Get choice from user

Choice:-1 calls add new records

Choice:-2 calls view records

Choice:-3 search employee

Choice:-4 calls modify or edit records

Choice:-5 calls delete or remove records

Choice:-6 calls calculate salary

Choice:-7 calls calculate bonus

Choice:-8 about

Choice:-0 exit

**Case 1: Add new records**

**Steps**

1.Declare file pointer ‘fp’

2.Display categories of information to be added of an employee

3.Get option from user .Does user want to add employee information

y:-goto step 3

n:-go back to main menu

4. Open file ‘fp’ to write

5. Assign the pointer to the end of the file to write

6. Get data from user

7. Write input data on a file

8. Close file

9.Print option to add another employee record

y:-goto step 1

n:-go back to main menu

**Case 2: View records**

**Steps**

1. Open ‘ftp’ file
2. Assign the pointer to the beginning of the file to be read
3. Loop until ‘End of file’ is not encountered read data from file
4. Display list of all employee with complete information
5. Calculate salary. Ask user does he want to calculate salary.
6. Close file

Go back to main menu

**Case 3: Search employee**

**Steps**

1. Enter the employee id from user

2. View the employee details.

**Case 4: Modify or edit records**

**Steps**

1. Enter employee id from user
2. Open file ‘ftp’
3. Assign the pointer to the beginning of the file to be read
4. Loop until ‘End of file ‘ is not encountered read data from file
5. a.Get new data from user of that employee which to be modified

b. Assign the pointer to the current position

c. Overwrite the new data on old data of that employee and goto step 7

n:- Display sorry message and goto step 7

1. Close file
2. Display option to edit another employee

y:-goto step 1

n:-goto step 8

1. Go back to main menu

**Case 5: Delete or modify records**

**Steps**

1. Enter employee name from user to delete

2. Declare file pointer ‘fpr’

3. Open file ‘ftp’ to read

4. Assign the pointer to the beginning of the file to be read

5. Loop until ‘End of file’ is not encountered read data from file.

6. Is user input employee name= employee name on a file

y:- a. open ‘fpr’ file and copy all data of ‘ftp’ file in ‘fpr’file except that data which we want to delete

b. delete ‘fpr’ file and rename ft file by ‘fpr’ file name and goto step 7

no:-print error message and close file and go back to main menu

7.close file

8.print option to delete another employee information

y:-goto step1

n:-go back to main menu

**Case 6: Calculate salary**

**Steps**

1.Enter employee id from the user to calculate salary.

2.Then enter the month and his attendance of this month.

3.Display shows the salary of the entered id.

**Case 7: Calculate bonus**

**Steps**

1. Enter employee id from the user to calculate bonus.
2. Then enter his/her grade, current date, month & year.
3. Display shows then the bonus of the entered id.

**Case 8: About**

**Steps**

1. Name of the developers and their institution.
2. Details of this project.

**Password function**

**Steps**

1. Declare two string ‘pass’ and ‘ password=bubt293’

2. Print “Please enter password”.

1. Until user entered the enter key get character for user
2. Store input character in string ‘pass’
3. Compare two string ‘pass’ and ‘password’ .Is both equal

Yes:-goto step 6

No:-print error message and goto step 2

6. Print password match message

7. Goto main menu

# Source code

//list of header files

#include<windows.h>

#include<stdio.h>

#include<conio.h>

#include <stdlib.h>

#include<string.h> //contains strcmp(),strcpy(),strlen(),etc

//#include<ctype.h> //contains toupper(), tolower(),etc

//#include<dos.h> //contains \_dos\_getdate

//#include<time.h>

COORD cursor = {0,0};

void gotopoint(int x,int y)

{

cursor.X = x;

cursor.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),cursor);

}

void Password();

int main()

{

Password();

getche();

return 0;

}

void mainmenu()

{

FILE \*fp, \*ftp, \*fpr;

char another, choice;

struct emp

{ char id[40];

char name[40];

int age;

float basic;

char address[100];

int dd,mm,yy;

char peradd[100];

double salary;

double present;

};

struct emp e;

char empid[40];

char ch;

int j,i=0,k,p;

int cas=0,cnt,dt=0,bon,modi,search=0,l,idlen=0,modilen1=0,modilen2=0,correct;

char design1[40]={"Employee Record System"};

char design2[10000]={" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*EMPLOYEE RECORD LIST\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"};

char xt[100]={"---\aThank You---"};

char catg;

char mnth[100];

char rafi[100]={"1. DEVELOPERS :"};

char dev1[100]={"i. Rafiqul Islam Rafi"};

char dev2[100]={"ii. Mohammad Yeasin Arafat"};

char dev3[100]={"iii. Syeda Farhana Aziz"};

char dev4[100]={"iv. Fazley Rabbi Ripon"};

char dev5[100]={"v. Marjia Akter Mita"};

char bubt[200]={"-----> Bangladesh University of Business and Technology(BUBT)"};

char about[100]={"2.Details :"};

char details[1000]={"The project is made to record employee details.\nIts also used to search recorded employee and calculate\n salary and bonus.The project is made by using c language..."};

long int size;

int day,month,year;

fp = fopen("EMP.DAT","rb+");

if(fp == NULL)

{

fp = fopen("EMP.DAT","wb+");

if(fp == NULL)

{

printf("Can't open file");

exit(1);

}

}

size = sizeof(e);

while(1)

{

system("cls");

gotopoint(6,1);

for(k=0;k<21;k++)

{

Sleep(40);

if(k%2==0)

printf("-");

else

printf("-");

}

for(k=0;k<strlen(design1);k++)

{

Sleep(40);

printf("%c",design1[k]);

}

for(k=0;k<21;k++)

{

Sleep(40);

if(k%2==0)

printf("-");

else

printf("-");

}

gotopoint(30,3);

printf("1. Add New Records");

gotopoint(30,5);

printf("2. View Records");

gotopoint(30,7);

printf("3. Search employee");

gotopoint(30,9);

printf("4. Modify or Edit or Update Records");

gotopoint(30,11);

printf("5. Delete or Remove Records");

gotopoint(30,13);

printf("6. Calculate salary");

gotopoint(30,15);

printf("7. Calculate bonus");

gotopoint(30,17);

printf("8. About");

gotopoint(30,19);

printf("0. Exit");

printf("\n");

gotopoint(20,21);

gotopoint(20,25);

printf("Please Enter Your Choice: ");

fflush(stdin); // flush the input buffer

choice = getche();

switch(choice)

{

case '1':

system("cls");

fseek(fp,0,SEEK\_END);

another = 'y';

while(another == 'y')

{

printf("\n Enter employee id: ");

again:

scanf("%[^\n]",&e.id);

idlen=strlen(e.id);

for(p=0;p<idlen;p++)

{

if((e.id[p]>='0'&&e.id[p]<='9'))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter id :");

goto again;

break;

}

}

printf("\n Enter Name: ");

again1:

scanf("%s",e.name);

l=strlen(e.name);

for(i=0;i<l;i++)

{

if((e.name[i]>='a'&&e.name[i]<='z')||(e.name[i]>='A'&&e.name[i]<='Z')||(e.name[i]==' '))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter name :");

goto again1;

break;

}

}

printf("\n Enter Age: ");

scanf("%d", &e.age);

if(e.age<=19||e.age>=89||(e.age>='a'&&e.age<='z')||(e.age>='A'&&e.age<='Z'))

{printf("Invalid age\n");

printf("\n\nPress any key to go Main Menu\n");

getch();

break;

}

printf("\n Enter Basic Salary: ");

scanf("%f", &e.basic);

printf("\n Enter Address: ");

scanf("%s",e.address);

printf("\n Enter Permanent Address: ");

scanf("%s",e.peradd);

printf("\n Enter Joining Date format (day month year): ");

scanf("%d%d%d",&e.dd,&e.mm,&e.yy);

if((e.dd>=1&&e.dd<=31)&&(e.mm>=1&&e.mm<=12))

{

printf("\n\n\nSuccessfully added\n\n");

}

else

{

printf("Invalid date or month\n");

printf("\n\nPress any key to go Main Menu\n");

getch();

break;

}

fwrite(&e,size,1,fp);

printf("\nAdd another record(y/n) ");

fflush(stdin);

another = getche();

}

break;

case '2':

system("cls");

rewind(fp);

// gotopoint(5,5);

printf("%s",design2);

gotopoint(0,3);

printf("Name");

gotopoint(15,3);

printf("id");

gotopoint(25,3);

printf("Age");

gotopoint(35,3);

printf("Salary");

gotopoint(45,3);

printf("Address");

gotopoint(60,3);

printf("Permanent Address");

gotopoint(85,3);

printf("Joining Date");

cas=3;

while(fread(&e,size,1,fp)==1)

{

cas+=2;

gotopoint(0,cas);

printf("%s",e.name);

gotopoint(15,cas);

printf("%s",e.id);

gotopoint(25,cas);

printf("%d",e.age);

gotopoint(35,cas);

printf("%0.2lf",e.basic);

gotopoint(45,cas);

printf("%s",e.address);

gotopoint(60,cas);

printf("%s",e.peradd);

gotopoint(85,cas);

printf("%d-%d-%d",e.dd,e.mm,e.yy);

}

printf("\n\n\n\nDo u want to Calculate Salary ?(y/n)\n");

another=getche();

getche();

while(another=='y')

{

double temp=0,d,present;

printf("\nEnter id of Employee to Calculate Salary: ");

scanf("%s",empid);

fpr = fopen("Temp2.dat","wb");

rewind(fp);

//if(strcmp(e.id,empid) != 0)

while(fread(&e,size,1,fp) == 1)

{

if(strcmp(e.id,empid) == 0){

printf("\n Enter present month(january...december)\n");

scanf("%s",mnth);

cnt=1;

if(strcmp(mnth,"january")==0 || strcmp(mnth,"January")==0 || strcmp(mnth,"JANUARY")==0 || strcmp(mnth,"march")==0 || strcmp(mnth,"March")==0 || strcmp(mnth,"MARCH")==0 || strcmp(mnth,"may")==0 || strcmp(mnth,"May")==0 || strcmp(mnth,"MAY")==0 || strcmp(mnth,"august")==0 || strcmp(mnth,"August")==0 || strcmp(mnth,"AUGUST")==0 || strcmp(mnth,"july")==0 || strcmp(mnth,"July")==0 || strcmp(mnth,"JULY")==0 || strcmp(mnth,"october")==0 || strcmp(mnth,"October")==0 || strcmp(mnth,"OCTOBER")==0 || strcmp(mnth,"december")==0 || strcmp(mnth,"December")==0 || strcmp(mnth,"DECEMBER")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"february")==0 || strcmp(mnth,"February")==0 || strcmp(mnth,"FEBRUARY")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>28 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/28);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"april")==0 || strcmp(mnth,"April")==0 || strcmp(mnth,"APRIL")==0 || strcmp(mnth,"june")==0 || strcmp(mnth,"June")==0 || strcmp(mnth,"JUNE")==0 || strcmp(mnth,"september")==0 || strcmp(mnth,"September")==0 || strcmp(mnth,"SEPTEMBER")==0 || strcmp(mnth,"november")==0 || strcmp(mnth,"November")==0 || strcmp(mnth,"NOVEMBER")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>30 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/30);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

}

else

{

cnt=0;

continue;

}

}

if(cnt==0){

printf("\nIncorrect id....\n");

}

fclose(fpr);

printf("\nCalculate Another Employee Salary(y/n)");

fflush(stdin);

another = getche();

}

printf("\n\nPress any key to go Main Menu\n");

getch();

break;

case '3':

system("cls");

another = 'y';

cas=3;

while(another == 'y')

{

printf("\nEnter id of Employee to search: ");

again3:

scanf("%s",empid);

for(p=0;p<strlen(empid);p++)

{

if((empid[p]>='0'&&empid[p]<='9'))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter id :");

goto again3;

break;

}

}

cas+=3;

rewind(fp);

while(fread(&e,size,1,fp) == 1)

{

if(strcmp(e.id,empid) == 0)

{ gotopoint(0,cas);

printf("Name");

gotopoint(15,cas);

printf("id");

gotopoint(25,cas);

printf("Age");

gotopoint(35,cas);

printf("Salary");

gotopoint(45,cas);

printf("Address");

gotopoint(60,cas);

printf("Permanent Address");

gotopoint(85,cas);

printf("Joining Date");

cas+=2;

gotopoint(0,cas);

printf("%s",e.name);

gotopoint(15,cas);

printf("%s",e.id);

gotopoint(25,cas);

printf("%d",e.age);

gotopoint(35,cas);

printf("%0.2lf",e.basic);

gotopoint(45,cas);

printf("%s",e.address);

gotopoint(60,cas);

printf("%s",e.peradd);

gotopoint(85,cas);

printf("%d-%d-%d",e.dd,e.mm,e.yy);

search=1;

break;

}

if(strcmp(e.id,empid)!=0)

{

search=0;

}

}

if(search==0)

{

printf("\n\nSorry! The record not found\n");

cas+=3;

}

search=0;

printf("\n\ndo u want to search another record (y/n) ? : ");

another = getche();

getche();

cas+=3;

if(another=='n')

{

printf("\n\n\n\n\nenter any key for main menu\n");

getche();

}

}

break;

case '4':

system("cls");

another = 'y';

while(another == 'y')

{

printf("Enter the Employee id to Modify: ");

again4:

scanf("%s", empid);

for(p=0;p<strlen(empid);p++)

{

if((empid[p]>='0'&&empid[p]<='9'))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter id :");

goto again4;

break;

}

}

rewind(fp);

while(fread(&e,size,1,fp)==1)

{

if(strcmp(e.id,empid) == 0){

printf("\n\nRecord found and Able to modify\n\n");

printf("\n Enter new id : ");

idmodi:

scanf("%s",e.id);

modilen2=strlen(e.id);

for(p=0;p<modilen2;p++)

{

if((e.id[p]>='0'&&e.id[p]<='9'))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter id :");

goto idmodi;

break;

}

}

printf("\n Enter Name : ");

namemodi:

scanf("%s",e.name);

modilen1=strlen(e.name);

for(i=0;i<modilen1;i++)

{

if((e.name[i]>='a'&&e.name[i]<='z')||(e.name[i]>='A'&&e.name[i]<='Z')||(e.name[i]==' '))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter name :");

goto namemodi;

break;

}}

printf("\n Enter Age: ");

scanf("%d", &e.age);

if(e.age<=19||e.age>=89||(e.age>='a'&&e.age<='z')||(e.age>='A'&&e.age<='Z'))

{printf("Invalid age\n");

printf("\n\nPress any key to go Main Menu\n");

getch();

break;

}

printf("\n Enter Basic Salary: ");

scanf("%f", &e.basic);

printf("\n Enter Address: ");

scanf("%s",e.address);

printf("\n Enter Permanent Address: ");

scanf("%s",e.peradd);

printf("\n Enter Joining Date format (day month year): ");

scanf("%d%d%d",&e.dd,&e.mm,&e.yy);

if((e.dd>=1&&e.dd<=31)&&(e.mm>=1&&e.mm<=12))

{

correct=0;

}

else

{

printf("Invalid date or month\n");

printf("\n\nPress any key to go Main Menu\n");

getch();

break;

}

fseek(fp,-size,SEEK\_CUR);

fwrite(&e,size,1,fp);

printf("\n\n successfully modified.....\n");

modi=1;

break;

}

else

modi==0;

}

if((strcmp(e.id,empid)!=0) && modi!=1)

{

printf("\aSorry ! Record not Found");

printf("\n\nPRESS ANY KEY FOR MAIN MENU\n");

}

if(strcmp(e.id,empid) == 0)

printf("\nModify another record(y/n)");

fflush(stdin);

another = getche();

}

break;

case '5':

system("cls");

another = 'y';

dt=0;

while(another == 'y')

{

printf("\nEnter id of Employee to Delete: ");

delid:

scanf("%s",empid);

for(p=0;p<strlen(empid);p++)

{

if((empid[p]>='0'&&empid[p]<='9'))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter id :");

goto delid;

break;

}

}

ftp = fopen("Temp.dat","wb");

rewind(fp);

while(fread(&e,size,1,fp) == 1)

{

if(strcmp(e.id,empid) != 0)

{

fwrite(&e,size,1,ftp);

}

if(strcmp(e.id,empid)==0)

{

printf("\n\n\nThe record is successfully deleted\n\n\n");

dt=1;

}

}

fclose(fp);

fclose(ftp);

remove("EMP.DAT");

rename("Temp.dat","EMP.DAT");

fp = fopen("EMP.DAT", "rb+");

if(dt==0)

{

printf("\n\n\aThe record not found\n\n\n");

}

dt=0;

printf("\nDelete another record(y/n)");

fflush(stdin);

another = getche();

}

break;

case '6':

system("cls");

another='y';

while(another=='y')

{

double temp=0,d,present;

printf("\nEnter id of Employee to Calculate Salary: ");

salid:

scanf("%s",empid);

for(p=0;p<strlen(empid);p++)

{

if((empid[p]>='0'&&empid[p]<='9'))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter id :");

goto salid;

break;

}

}

fpr = fopen("Temp2.dat","wb");

rewind(fp);

//if(strcmp(e.id,empid) != 0)

while(fread(&e,size,1,fp) == 1)

{

if(strcmp(e.id,empid) == 0){

printf("\n Enter present month(january...december)\n");

scanf("%s",mnth);

cnt=1;

if(strcmp(mnth,"january")==0 || strcmp(mnth,"January")==0 || strcmp(mnth,"JANUARY")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"february")==0 || strcmp(mnth,"February")==0 || strcmp(mnth,"FEBRUARY")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>28 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/28);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"march")==0 || strcmp(mnth,"March")==0 || strcmp(mnth,"MARCH")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"april")==0 || strcmp(mnth,"April")==0 || strcmp(mnth,"APRIL")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>30 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/30);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"may")==0 || strcmp(mnth,"May")==0 || strcmp(mnth,"MAY")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"june")==0 || strcmp(mnth,"June")==0 || strcmp(mnth,"JUNE")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>30 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/30);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"july")==0 || strcmp(mnth,"July")==0 || strcmp(mnth,"JULY")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"august")==0 || strcmp(mnth,"August")==0 || strcmp(mnth,"AUGUST")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"september")==0 || strcmp(mnth,"September")==0 || strcmp(mnth,"SEPTEMBER")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>30 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/30);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"october")==0 || strcmp(mnth,"October")==0 || strcmp(mnth,"OCTOBER")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"november")==0 || strcmp(mnth,"November")==0 || strcmp(mnth,"NOVEMBER")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>30 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/30);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else if(strcmp(mnth,"december")==0 || strcmp(mnth,"December")==0 || strcmp(mnth,"DECEMBER")==0){

printf("\nEnter his/her attendence of this month:\n");

scanf("%lf",&present);

if(present>31 || present<0)

{

printf("\ninvalid day\n");

cnt=1;

break;

}

temp=(e.basic/31);

printf("\n%s will get %0.2lf taka of this month",e.name,temp\*present);

cnt=1;

break;

}

else

{

printf("\ninvalid month\n");

break;

}

}

else

{

cnt=0;

continue;

}

}

if(cnt==0){

printf("\nIncorrect id....\n");

}

fclose(fpr);

printf("\nCalculate Another Employee Salary(y/n)");

fflush(stdin);

another = getche();

}

printf("\n\n\nPress any key to go Main Menu\n");

getch();

break;

case '7':

system("cls");

another='y';

while(another=='y')

{

printf("\n\nEnter id of Employee to Calculate bonus: ");

bonusid:

scanf("%s",empid);

for(p=0;p<strlen(empid);p++)

{

if((empid[p]>='0'&&empid[p]<='9'))

{

continue;

}

else

{ printf("You Entered wrong keyword!\n\nPlease again Enter id :");

goto bonusid;

break;

}

}

printf("\nEnter the Category of this Employee(A/B/C/D)\n: ");

catg=getche();

getche();

fpr = fopen("Temp2.dat","wb");

rewind(fp);

while(fread(&e,size,1,fp) == 1)

{

if(strcmp(e.id,empid) == 0){

printf("\nEnter Present date :\n");

scanf("%d",&day);

if(day>=32)

{printf("Invalid date\n");

printf("\n\nPress any key to go Main Menu\n");

getch();

break;

}

printf("\nEnter Present month :\n");

scanf("%d",&month);

if(month>=13)

{printf("Invalid month\n");

printf("\n\nPress any key to go Main Menu\n");

getch();

break;

}

printf("\nEnter Present year :\n");

scanf("%d",&year);

if(e.yy==year)

{

printf("\n%s will not get bonus in year %d\n",e.name,year);

break;

}

else if(year-e.yy==1)

{

if(month>=e.mm)

{

if(day>=e.dd&&catg=='A')

printf("\n%s will get =%0.2f taka as bonus in year %d\n",e.name,0.5\*e.basic,year);

if(day>=e.dd&&catg=='B')

printf("\n%s will get =%0.2f taka as bonus in year %d\n",e.name,0.25\*e.basic,year);

if(day>=e.dd&&catg=='C')

printf("\n%s will get =%0.2f taka as bonus in year %d\n",e.name,0.10\*e.basic,year);

if(day<e.dd&&catg=='D')

printf("\n%s will not get bonus in year %d\n",e.name,year);

}

if(month<e.mm)

printf("\n%s will not get bonus in year %d\n",e.name,year);

}

else if(year-e.yy>1){

if(catg=='A')

printf("\n%s will get =%0.2f taka as bonus in year %d\n",e.name,0.5\*e.basic,year);

if(catg=='B')

printf("\n%s will get =%0.2f taka as bonus in year %d\n",e.name,0.25\*e.basic,year);

if(catg=='C')

printf("\n%s will get =%0.2f taka as bonus in year %d\n",e.name,0.10\*e.basic,year);

if(catg=='D')

printf("\n%s will not get bonus in year %d\n",e.name,year);

}

else

{

printf("\n%s will not get bonus in year %d\n",e.name,year);

}

bon=1;

break;

}

else

{

bon=0;

continue;

}

}

if(bon==0)

printf("\nIncorrect employee id\n");

fclose(fpr);

printf("\nCalculate Another Employee bonus(y/n)\n");

fflush(stdin);

another = getche();

}

printf("\n\n\nPress any key to go Main Menu\n");

getch();

break;

case '8':

system("cls");

printf("%s",rafi);

gotopoint(6,1);

printf("%s",dev1);

gotopoint(6,3);

printf("%s",dev2);

gotopoint(6,5);

printf("%s",dev3);

gotopoint(6,7);

printf("%s",dev4);

gotopoint(6,9);

printf("%s",dev5);

gotopoint(12,11);

printf("%s",bubt);

gotopoint(0,15);

printf("%s",about);

gotopoint(1,17);

printf("%s",details);

printf("\n\n\nPress any key to go Main Menu\n");

getch();

break;

case '0':

fclose(fp);

gotopoint(20,30);

for(i=0;i<strlen(xt);i++)

{

Sleep(40);

printf("%c",xt[i]);

}

gotopoint(20,40);

exit(0);

}

}

}

void Password(void)

{

system("cls");

int i=0,j;

char ch;

char pass[1000];

char head1[50]={" System is Password Protected "};

char head2[50]={"Please Enter Password 3-8 character: "};

gotopoint(10,2);

for(j=0;j<20;j++)

{

Sleep(30);

if(j%2==0)

printf(">");

else

printf(">");

}

for(j=0;j<strlen(head1);j++)

{

Sleep(30);

printf("%c",head1[j]);

}

for(j=0;j<20;j++)

{

Sleep(30);

if(j%2==0)

printf("<");

else

printf("<");

}

gotopoint(16,7);

for(j=0;j<strlen(head2);j++)

{

Sleep(30);

printf("%c",head2[j]);

}

while(ch!=13) //ASCII value of enter is 13

{

ch=getch();

if(ch!=13 && ch!=8){

putch('\*');

pass[i] = ch;

i++;

}

}

pass[i] = '\0';

if(strcmp(pass,"bubt293")==0)

{

gotopoint(15,10);

printf("Welcome !Password match....");

gotopoint(15,13);

printf("Press any key to continue....");

getch();

mainmenu();

}

else

{

gotopoint(15,16);

printf("\a Sorry! You Entered Wrong Password....");

gotopoint(15,18);

printf("try again.....");

getch();

Password();

}

}

# Program Output

The screen below asks the user for password and if password matched then allow the user to go inside the program.

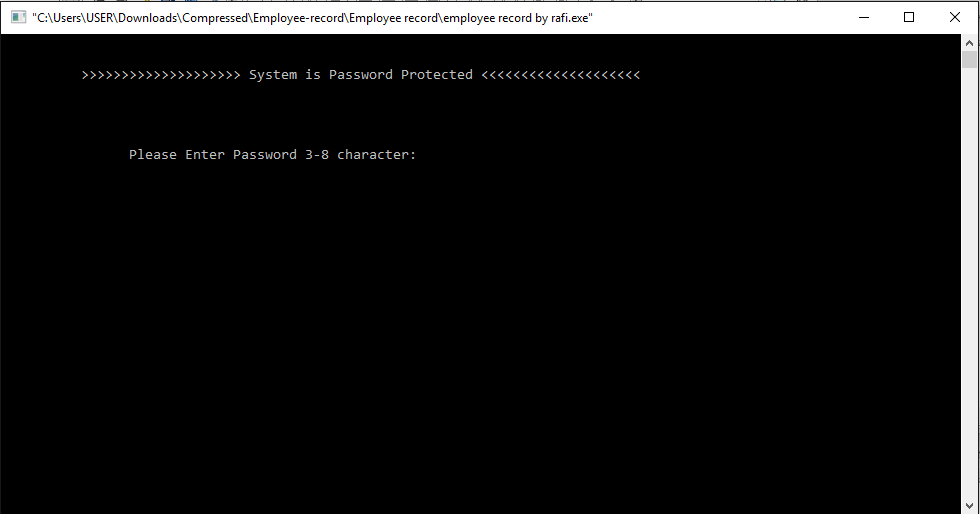


Fig:-1

Screen showing that the password is wrong

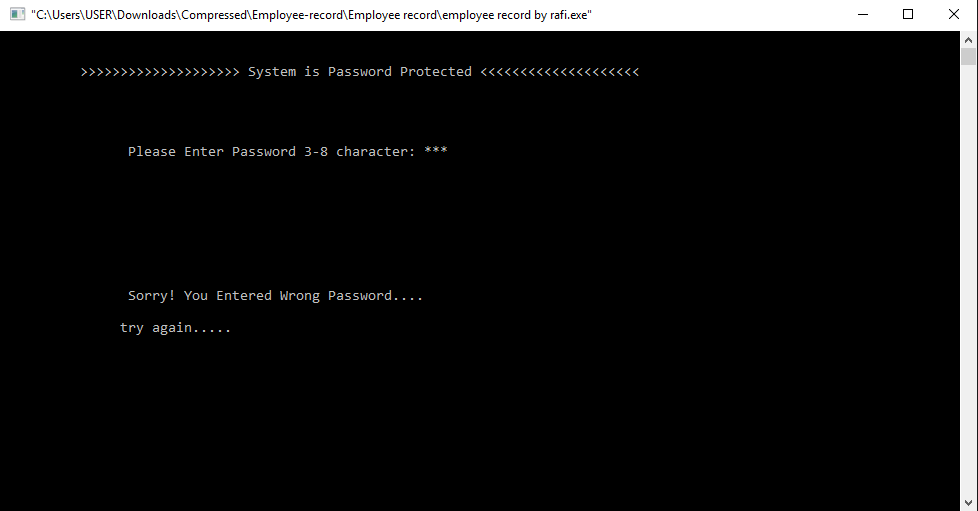


Fig:-2

Screen showing that the password is correct

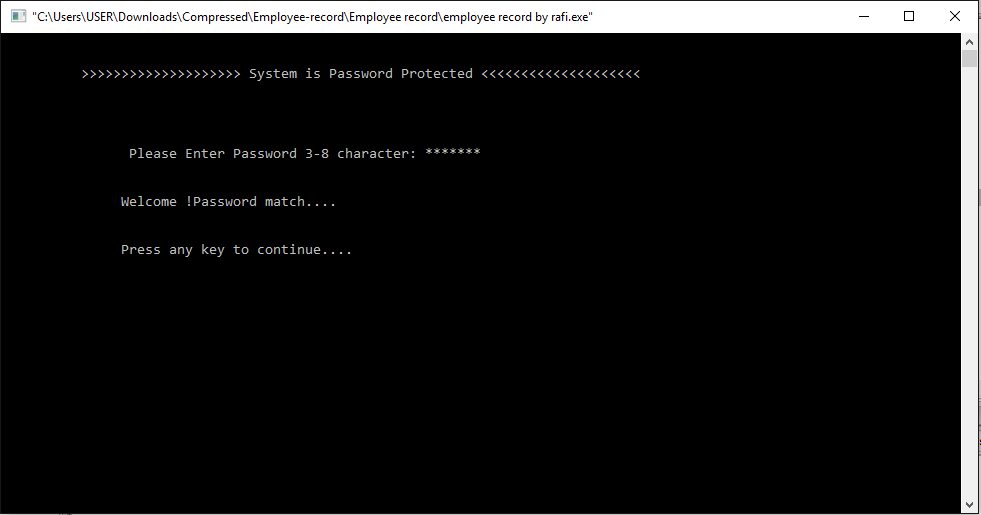


Fig:-3

The screen shows the style of main menu

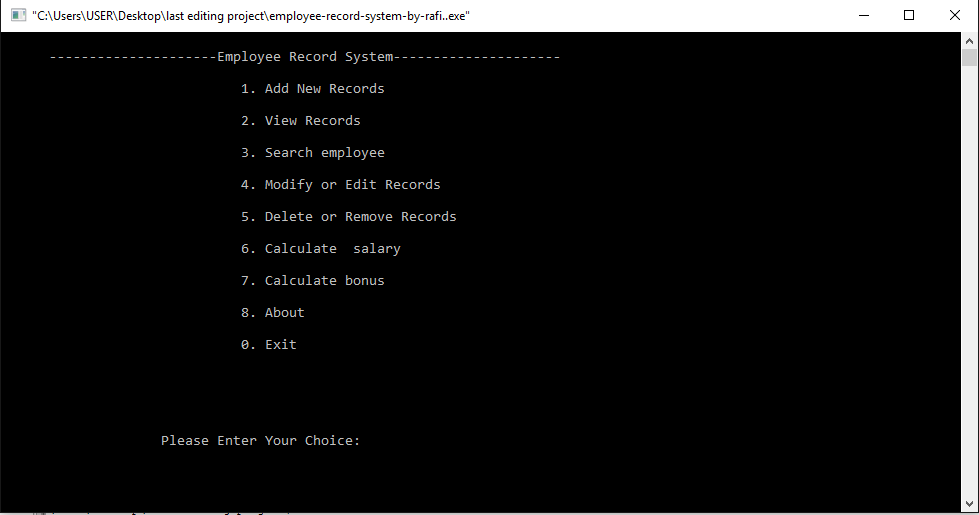


Fig:-4

Screen showing Adding and Saving Employee Record

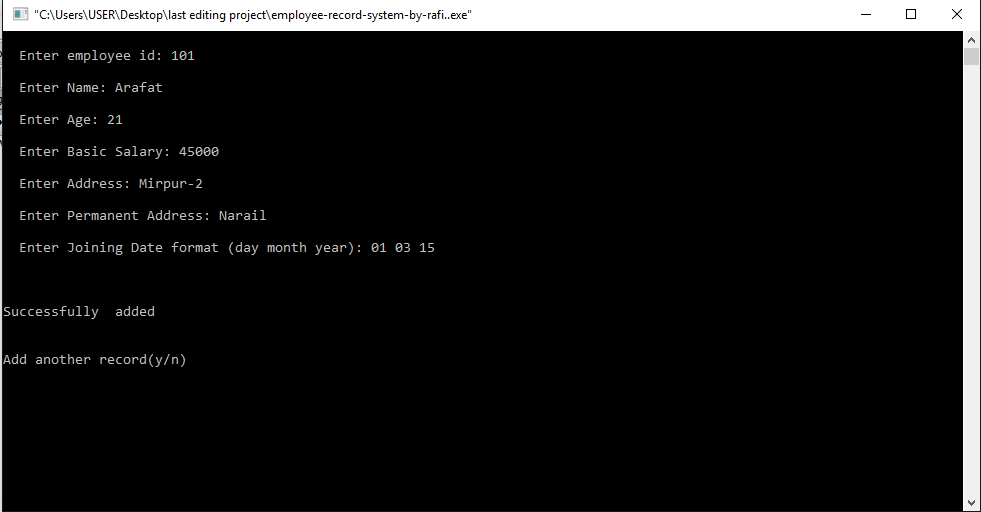


Fig:-5

Screen showing the List of Employee information

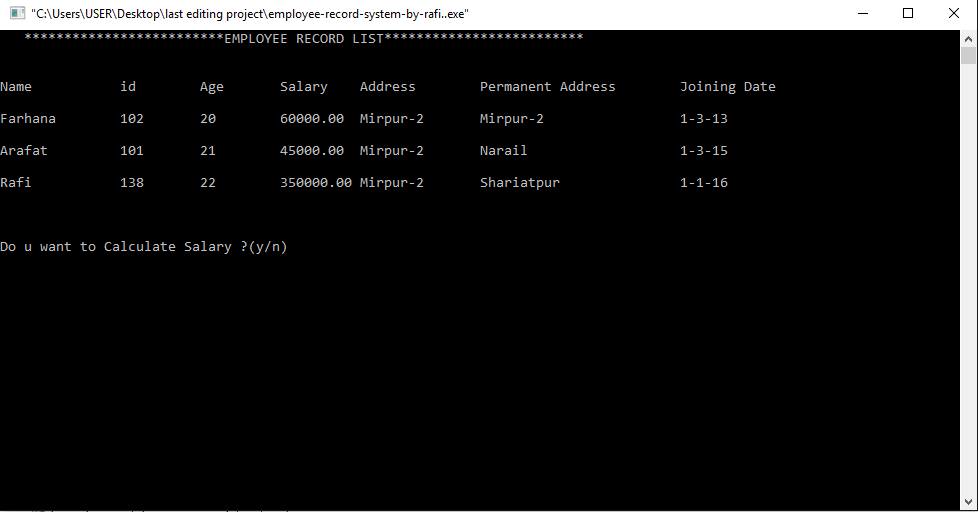


Fig:-6

Screen showing search employee details

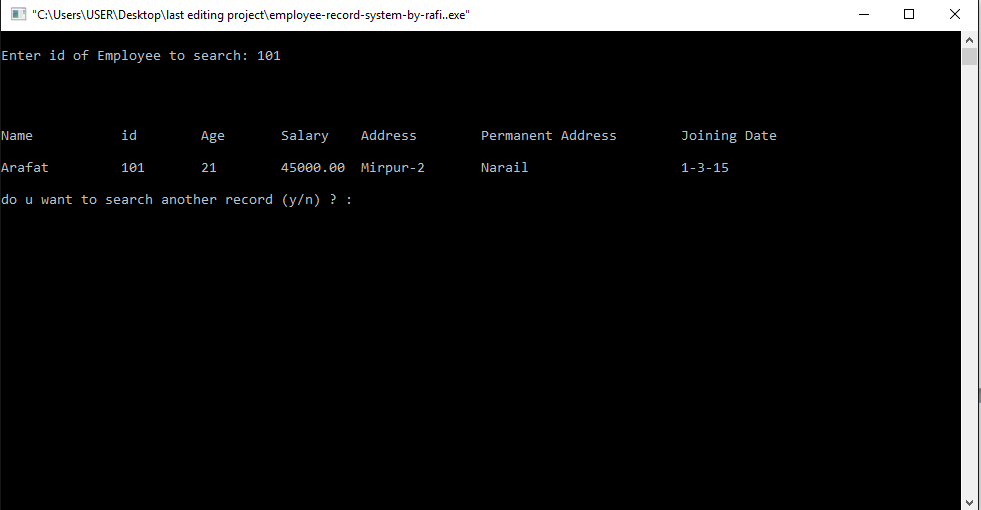


Fig:-7

Screen showing Modifying or Edit Employee Data

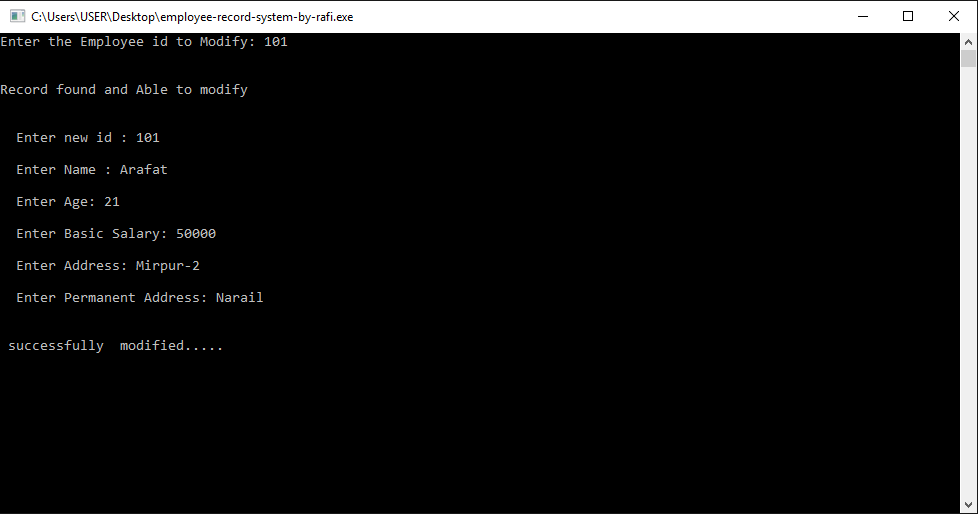


Fig:-8

Screen showing the Delete or Remove operation of Employee id

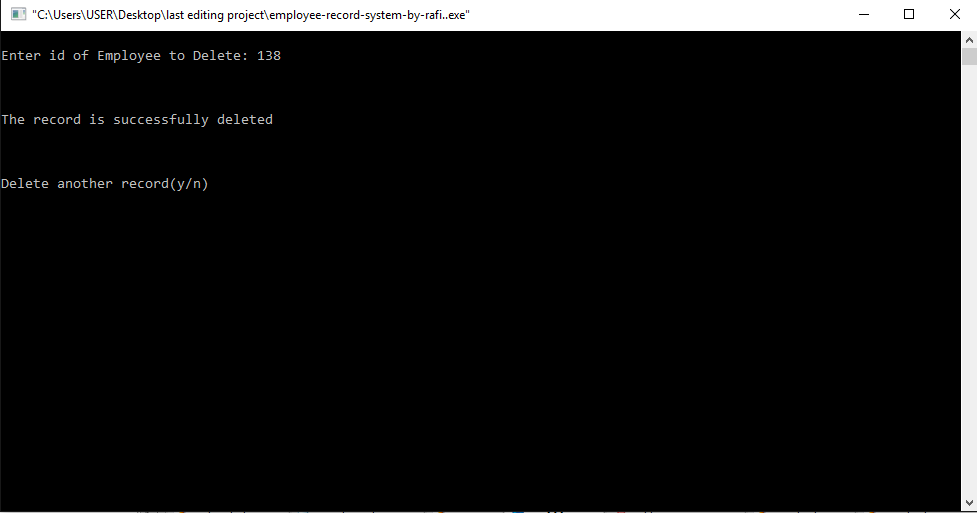


Fig:-9

Screen showing the Calculation of Salary

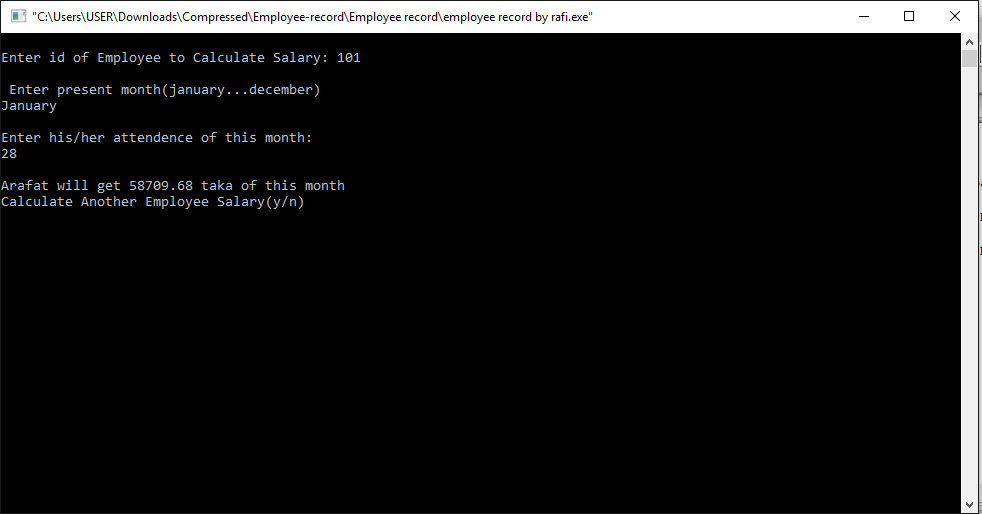


Fig:-10

Screen showing the Calculation of Bonus of Employee’s

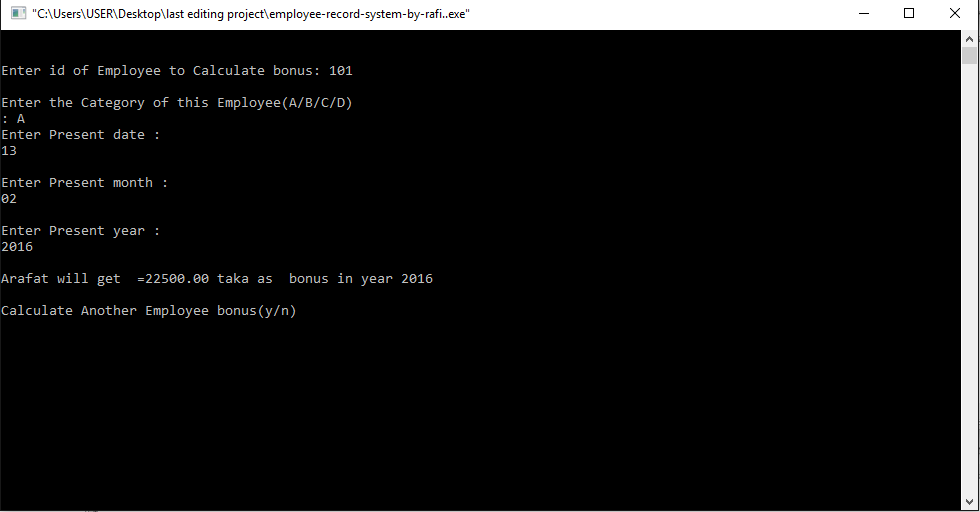


Fig:11

Screen showing the about operation

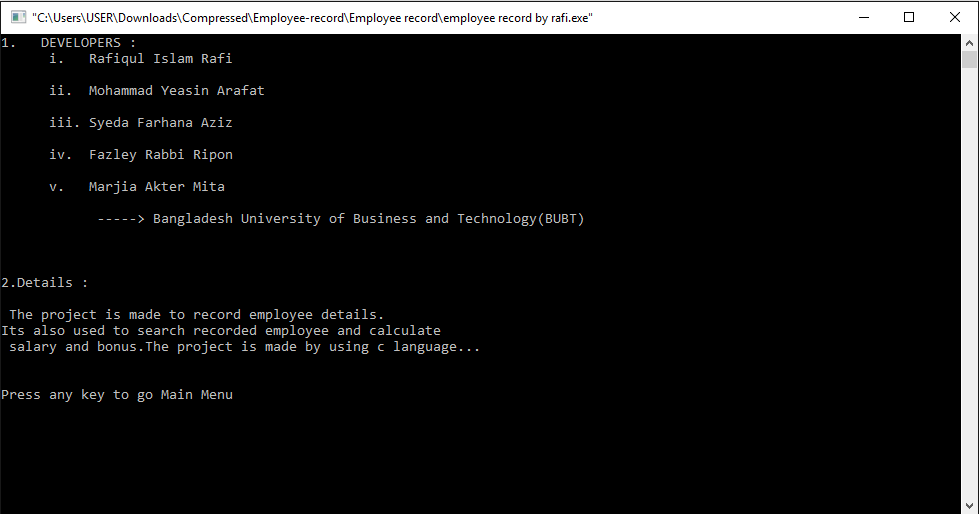


Fig:-12

Screen showing the Exit operation

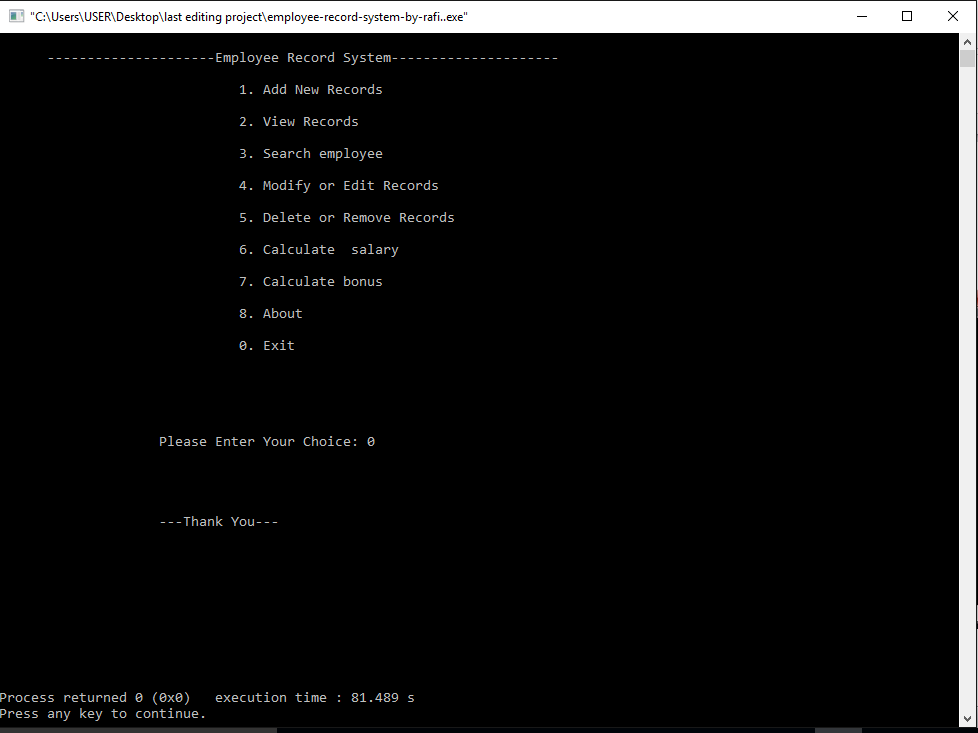


Fig:-13

# Discussion and Program Limitations

After a continuous and devoted attempt, we have finally completed our program ‘Employee Record System’. The program basically gives the idea about employee management, how to manage information about employees of a company. During the program execution we encountered tremendous problems which we are going to explain here.

We have divided our program into SEVEN division i.e. Add new records, View records, Search employee, Modify or edit records, Delete or remove records, Calculate salary and Calculate bonus. We try to handle our program by password option. But there is no option to change the password through the program. As we didn’t have any option for that in the program. This is one of the limitations of our program. In order to change the password we have to change it through the source code.

In add new records we add the information of the employee’s. We enter the id, name, age, address, joining date of the employee. If there is more than one employee with the same name, he/she can be differentiated with the employee id. This is one of the advantages of this program. We can add more than one employee data at a time.

In view records option, we show the list of the employee’s working in that company. We also have a calculate salary option there. If the user wants to calculate salary of any employee he/she can do that.

In search employee option, user searches an Id and shows the details of this Id.

In modify or edit records, we can modify the employee data of any employee. First the user has to enter the employee id which he/she wants to modify. If the id doesn’t exist then the process will not go forward. If the id exists then the user has to enter the new id then name, address etc. The there will be message saying that “THE RECORD IS SUCCESSFULLY SAVED”.

In delete or remove records option, the user should enter the Id of employee which he/she wants to delete. So the user must know the Id of employee which he wants to delete. If the program finds the employee, according to the Id entered by user, then it deletes it. If the user entered a wrong id, then there will be a message saying “PRESS ANY KEY TO GO TO MAIN MENU”.

We also have calculated salary and calculate bonus option. In calculate salary, we have to enter the id of the user which the user wants to find the salary. After the id is entered, then the present date is required to calculate the salary. The salary is calculated from the joining date.

The bonus is also calculated in the same way. If an employee is grade (A/B/C/D) define their bonus.

We write our program by the combination of many user defined function, calling them frequently when necessary. There may not be the proper arrangement of function and codes in the program. But tried our best to avoid above discussed limitation but still we think we are unable to do this. And we commit we will possibly avoid them in out coming projects.

# Summary

Hence after the completion of the project we got familiar with the C programming and its features.

A complete and useful employee record system can only be developed with lot of intensive effort and time. Due to lack of time and we are beginners in programming program that we expected can’t be developed by us. Our employee record system will be the most useful for study and programming practice using C.

As a whole, the project has been a good learning experience for us. We have gained knowledge about the various aspects of C programming. At the same time, we have developed a deep understanding about the file handling in C.

We still want to emphasize that the program is not complete by itself. There is still a lot of room for improvement. Graphics may be added to program to make it more attractive. The mouse cursor may be initialized in order to make the program even more interactive.