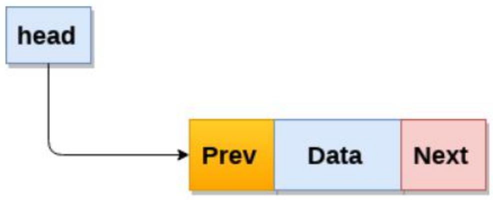
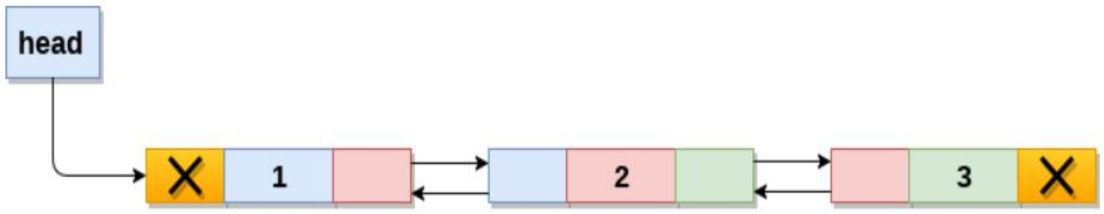
LITERATURE

Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence. Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer), pointer to the previous node (previous pointer). A sample node in a doubly linked list is shown in the figure.



HEAD



We have given Seven options to the user, named as –

1. Create list of all employees.
2. Insert new employee from beginning.
3. Insert new employee from end.
4. Search the employee
5. Delete the employee from beginning.
6. Delete the employee from end.
7. Display all employees.

For storing the data of the employee, create a user define datatype which will store the information regarding Employee. Below is the declaration of the data type:

```
struct node {
struct node* prev;
int ssn;
long int phno;
float sal;
char name[20], dept[10], desg[20];
struct node* next;}
```

Building the Employee's table:
For building the employee table the idea is to use the above struct datatype which will use to store the information regarding the employee and every new employee's details will be added as a linked list node.

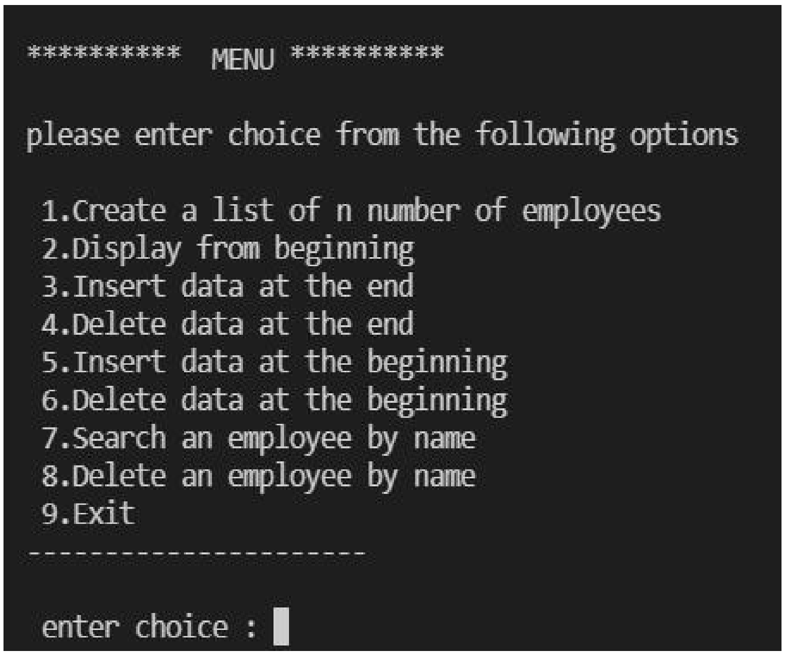
Deleting in the record:
Since, a doubly-linked list is used to store the data, therefore to delete the data at any index just link the next to the next of the deleted data and link the previous node of the next data of the deleted node to its previous data.

Searching in the record:
For searching in the record based on any parameter, the idea is to traverse the data and if at any index the value parameters match with the record stored, print all the information of that employee.

METHODOLOGY

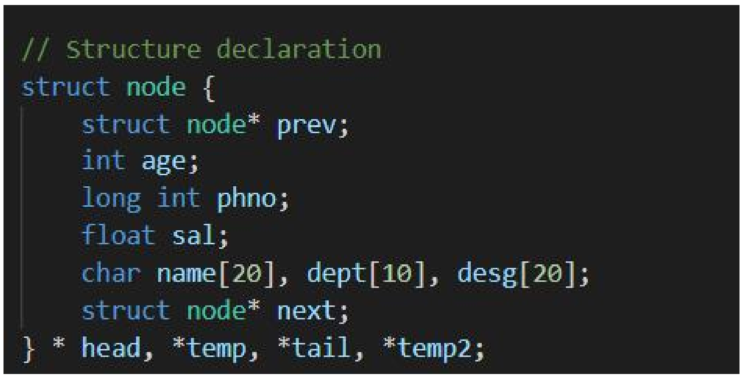
In this project, we are implementing an employee management system using a doubly linked list. This program displays a menu to the user and asks user to enter a choice from the following options:-

1.Create a list of n number of employees
2.Display from beginning
3.Insert data at the end
4.Delete data at the end
5.Insert data at the beginning
6.Delete data at the beginning
7.Search an employee by name
8.Delete an employee by name
9.Exit



```
**********  MENU **********

please enter choice from the following options

 1.Create a list of n number of employees
 2.Display from beginning
 3.Insert data at the end
 4.Delete data at the end
 5.Insert data at the beginning
 6.Delete data at the beginning
 7.Search an employee by name
 8.Delete an employee by name
 9.Exit
----------------------

 enter choice :
```

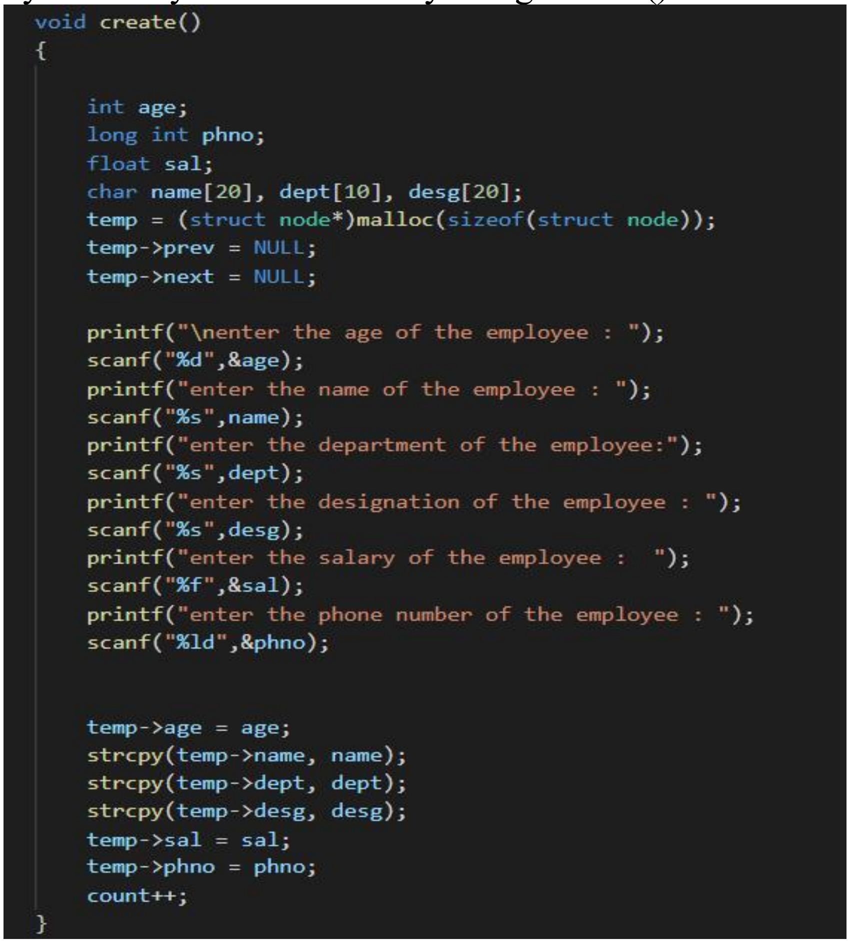
We have defined a structure named node with the following members:



```c
// Structure declaration
struct node {
    struct node* prev;
    int age;
    long int phno;
    float sal;
    char name[20], dept[10], desg[20];
    struct node* next;
} * head, *temp, *tail, *temp2;
```

CREATING A NEW NODE IN THE LIST:

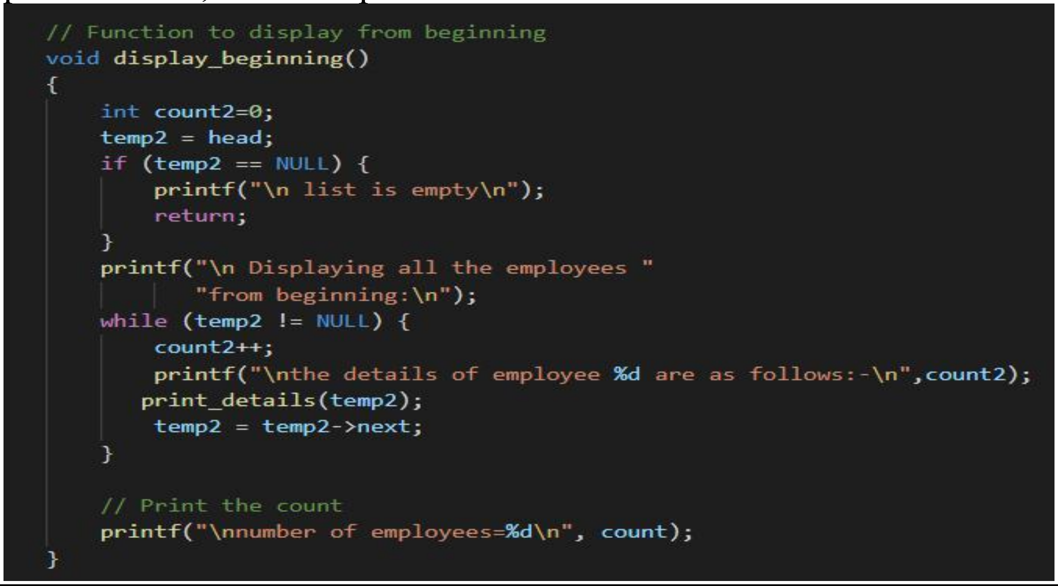
For creating a new node, we have defined a function named create() which will create a new node dynamically whenever required. A pointer named temp is used to point to dynamically allotted memory using malloc() function.



```c
void create()
{

    int age;
    long int phno;
    float sal;
    char name[20], dept[10], desg[20];
    temp = (struct node*)malloc(sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;

    printf("\nenter the age of the employee : ");
    scanf("%d",&age);
    printf("enter the name of the employee : ");
    scanf("%s",name);
    printf("enter the department of the employee:");
    scanf("%s",dept);
    printf("enter the designation of the employee : ");
    scanf("%s",desg);
    printf("enter the salary of the employee :  ");
    scanf("%f",&sal);
    printf("enter the phone number of the employee : ");
    scanf("%ld",&phno);


    temp->age = age;
    strcpy(temp->name, name);
    strcpy(temp->dept, dept);
    strcpy(temp->desg, desg);
    temp->sal = sal;
    temp->phno = phno;
    count++;
}
```

DISPLAYING THE LIST FROM BEGINNING:

To display the list of employees from beginning, we have defined a function named Display_beginning() in which we have created a pointer temp2 which will point to starting node only. After that, we iterate over the entire list and display the details of each node using print_details() function. We continue this process until the temp2 pointer points to null, which implies the list has ended.

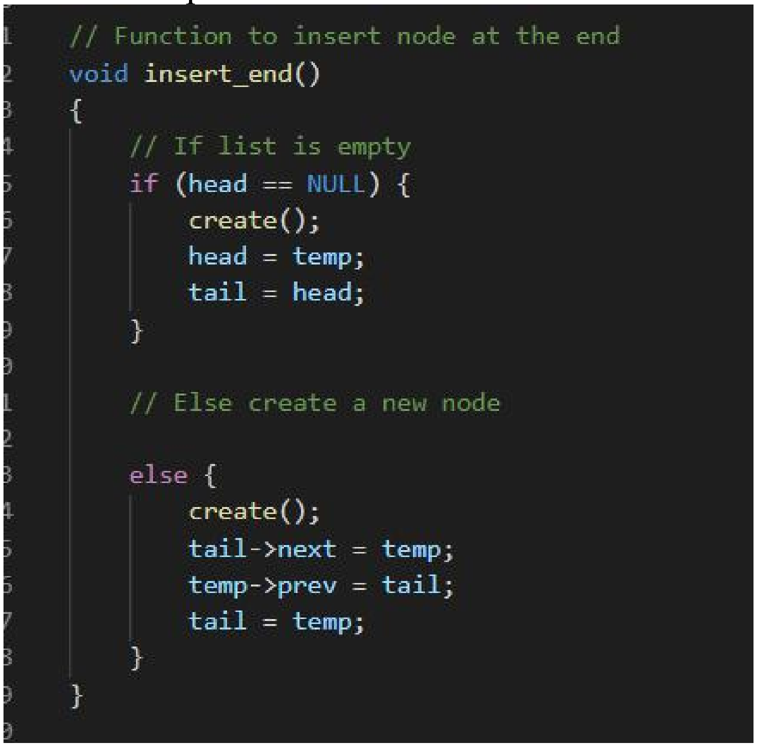


```c
// Function to display from beginning
void display_beginning()
{
    int count2=0;
    temp2 = head;
    if (temp2 == NULL) {
        printf("\n list is empty\n");
        return;
    }
    printf("\n Displaying all the employees "
           "from beginning:\n");
    while (temp2 != NULL) {
        count2++;
        printf("\nthe details of employee %d are as follows:-\n",count2);
       print_details(temp2);
        temp2 = temp2->next;
    }

    // Print the count
    printf("\nnumber of employees=%d\n", count);
}
```

INSERTING IN THE LIST:

Insert at the end:

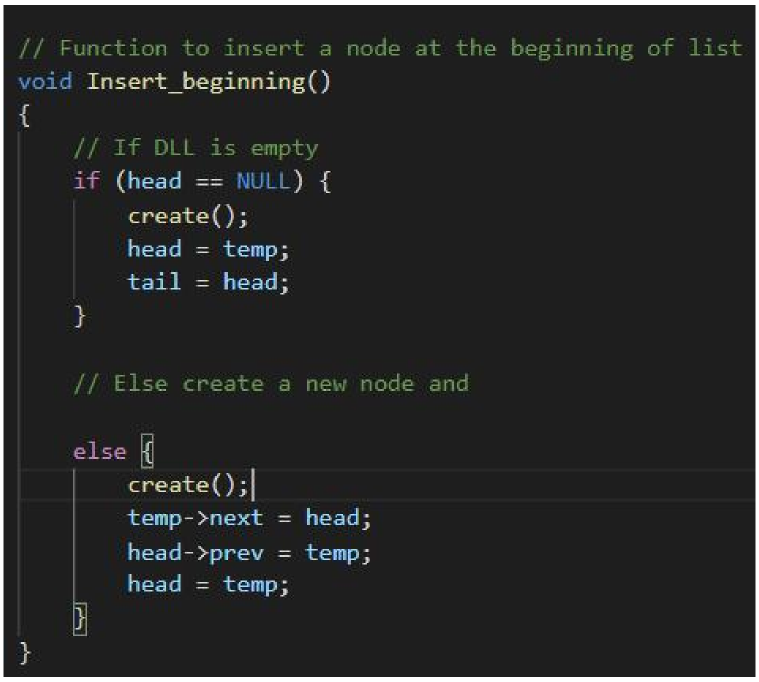
To perform insertion at the end of the list, we have defined a function named insert_end()
And this helps to create a new node in the list at the end.



```
// Function to insert node at the end
void insert_end()
{
    // If list is empty
    if (head == NULL) {
        create();
        head = temp;
        tail = head;
    }

    // Else create a new node

    else {
        create();
        tail->next = temp;
        temp->prev = tail;
        tail = temp;
    }
}
```

Insert at the beginning:

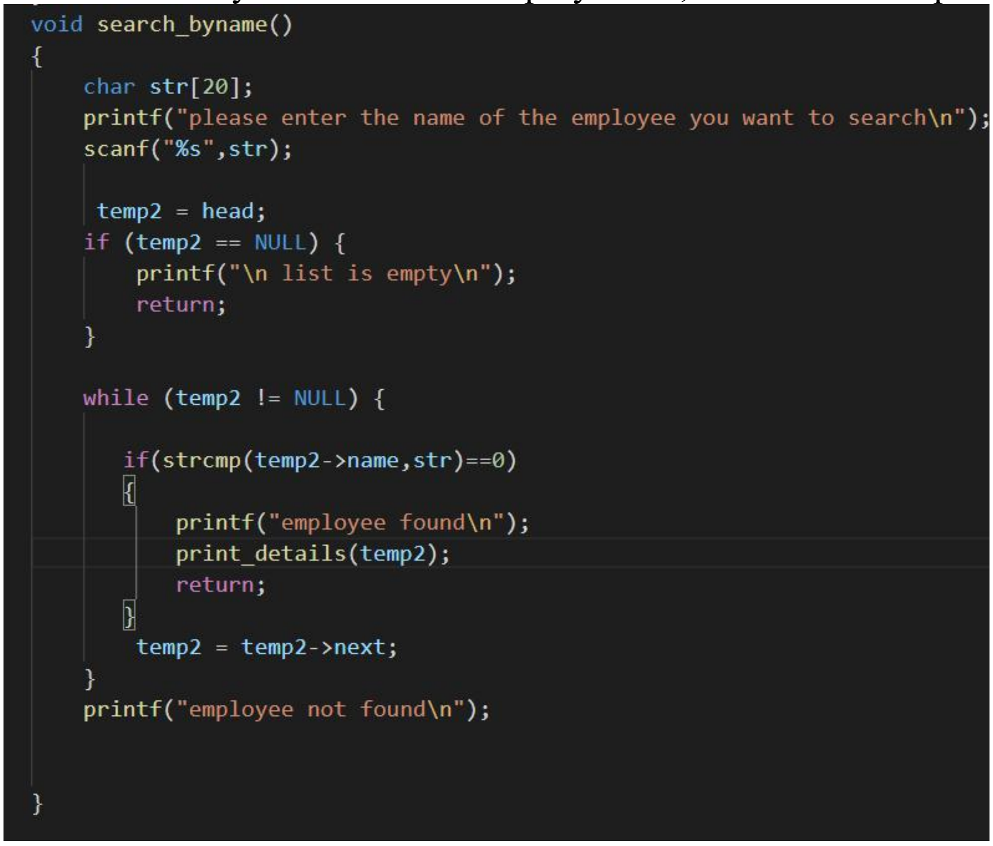
To perform insertion at the beginning of the list, we have defined a function named insert_beginning()
And this helps to create a new node in the list at the beginning.



```
// Function to insert a node at the beginning of list
void Insert_beginning()
{
    // If DLL is empty
    if (head == NULL) {
        create();
        head = temp;
        tail = head;
    }

    // Else create a new node and

    else {
        create();
        temp->next = head;
        head->prev = temp;
        head = temp;
    }
}
```

SEARCH:

In this function we have given the choice to the users so that they can search the employee from the employee list by calling his/her name. The function search_byname() then gives the information of the called employee. And if the function did not found the name entered by the user in the employee list, it shows the output as employee not found.

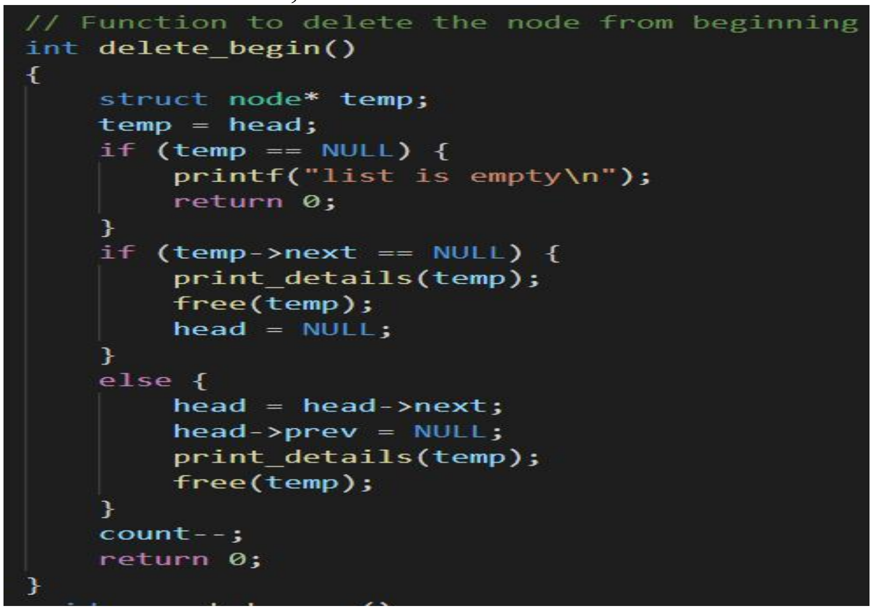


```c
void search_byname()
{
    char str[20];
    printf("please enter the name of the employee you want to search\n");
    scanf("%s",str);

     temp2 = head;
    if (temp2 == NULL) {
        printf("\n list is empty\n");
        return;
    }

    while (temp2 != NULL) {

       if(strcmp(temp2->name,str)==0)
       {
           printf("employee found\n");
           print_details(temp2);
           return;
       }
        temp2 = temp2->next;
    }
    printf("employee not found\n");


}
```

PERFORMING DELETION IN THE LIST:

DELETE FROM BEGINNING:

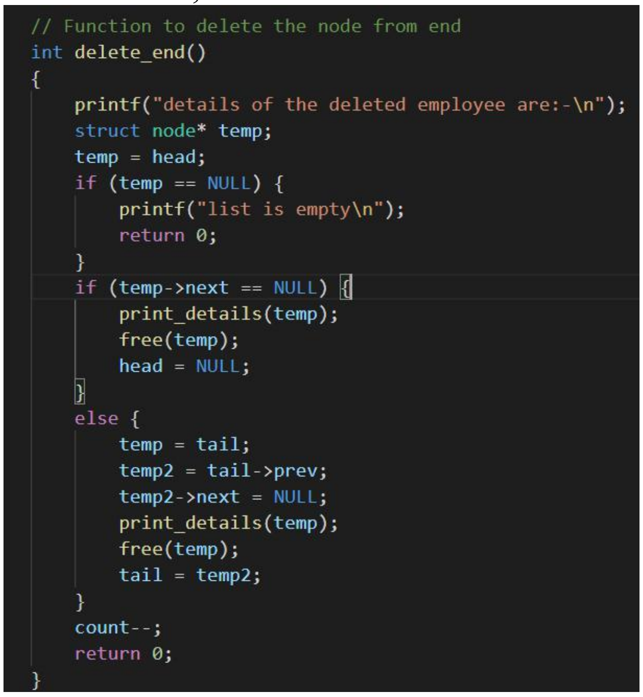
In this function we have given the choice to the users so that they can delete the information of the employee from the employee list from the beginning of the list. If user called this function, it first shows the information of the employee and then deletes it.



```c
// Function to delete the node from beginning
int delete_begin()
{
    struct node* temp;
    temp = head;
    if (temp == NULL) {
        printf("list is empty\n");
        return 0;
    }
    if (temp->next == NULL) {
        print_details(temp);
        free(temp);
        head = NULL;
    }
    else {
        head = head->next;
        head->prev = NULL;
        print_details(temp);
        free(temp);
    }
    count--;
    return 0;
}
```

DELETE FROM END:

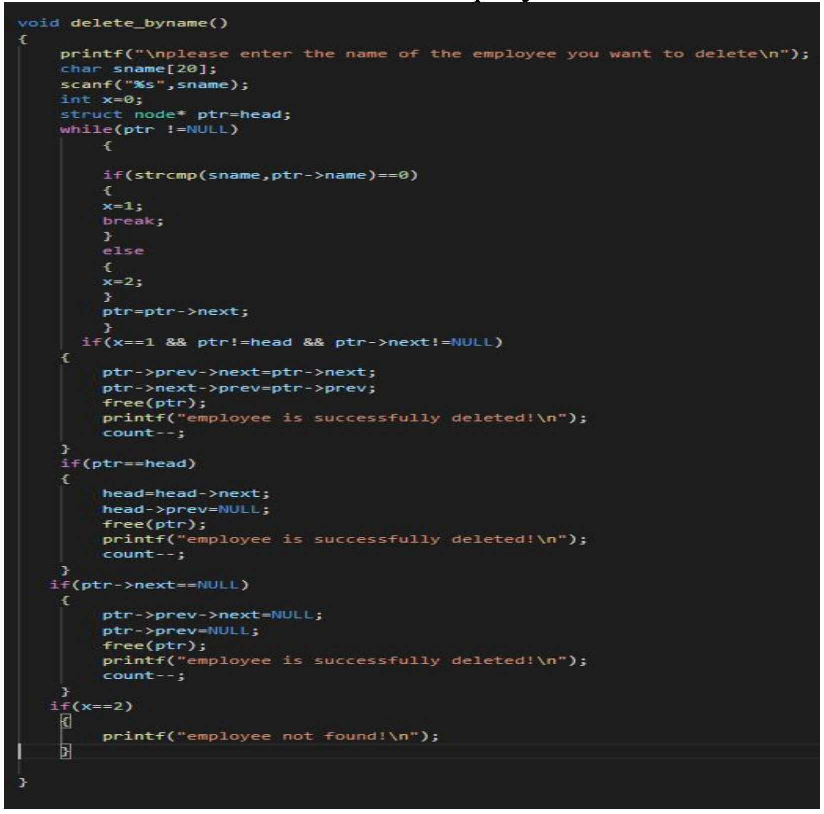
In this function we have given the choice to the users so that they can delete the information of the employee from the employee list from the end of the list. If user called this function, it first shows the information of the employee and then deletes it.



```c
// Function to delete the node from end
int delete_end()
{
    printf("details of the deleted employee are:-\n");
    struct node* temp;
    temp = head;
    if (temp == NULL) {
        printf("list is empty\n");
        return 0;
    }
    if (temp->next == NULL) {
        print_details(temp);
        free(temp);
        head = NULL;
    }
    else {
        temp = tail;
        temp2 = tail->prev;
        temp2->next = NULL;
        print_details(temp);
        free(temp);
        tail = temp2;
    }
    count--;
    return 0;
}
```

DELETE BY NAME :

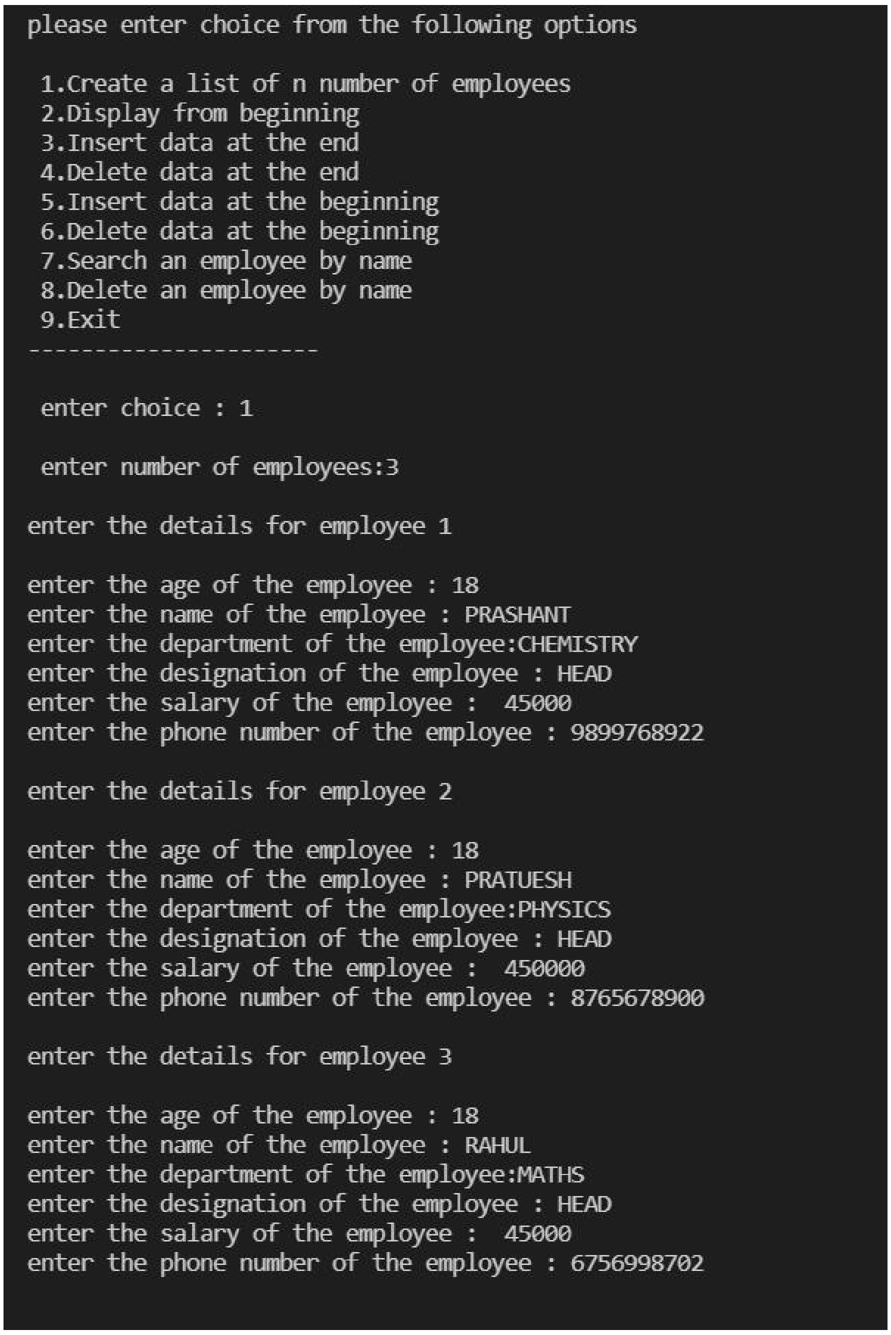
In this function we have given the choice to the users so that they can delete the information of the employee from the employee list by entering the name of the employee. If user called this function, it first ask the name of the employee which is to be deleted then after entering the name it shows the output as employee found and then it show the information of the employee and then deletes it from the list.



```c
void delete_byname()
{
    printf("\nplease enter the name of the employee you want to delete\n");
    char sname[20];
    scanf("%s",sname);
    int x=0;
    struct node* ptr=head;
    while(ptr !=NULL)
        {

        if(strcmp(sname,ptr->name)==0)
        {
        x=1;
        break;
        }
        else
        {
        x=2;
        }
        ptr=ptr->next;
        }
      if(x==1 && ptr!=head && ptr->next!=NULL)
    {
        ptr->prev->next=ptr->next;
        ptr->next->prev=ptr->prev;
        free(ptr);
        printf("employee is successfully deleted!\n");
        count--;
    }
    if(ptr==head)
    {
        head=head->next;
        head->prev=NULL;
        free(ptr);
        printf("employee is successfully deleted!\n");
        count--;
    }
   if(ptr->next==NULL)
    {
        ptr->prev->next=NULL;
        ptr->prev=NULL;
        free(ptr);
        printf("employee is successfully deleted!\n");
        count--;
    }
   if(x==2)
    {
        printf("employee not found!\n");
    }

}
```

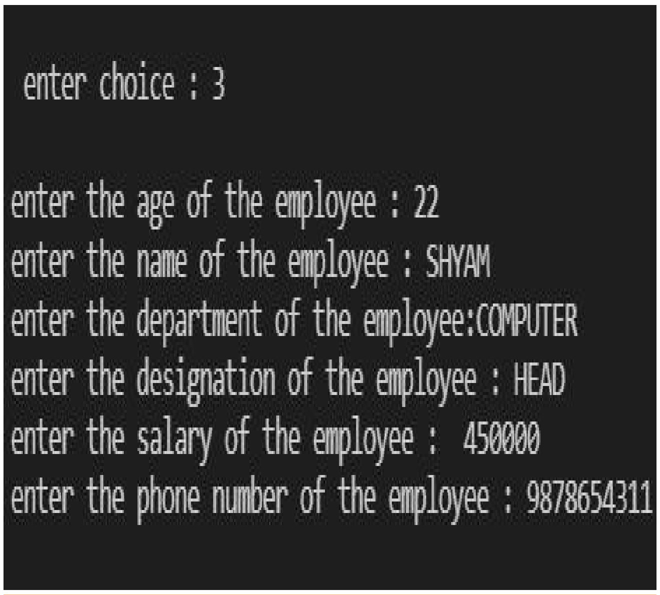
RESULT

- CREATING THE LIST



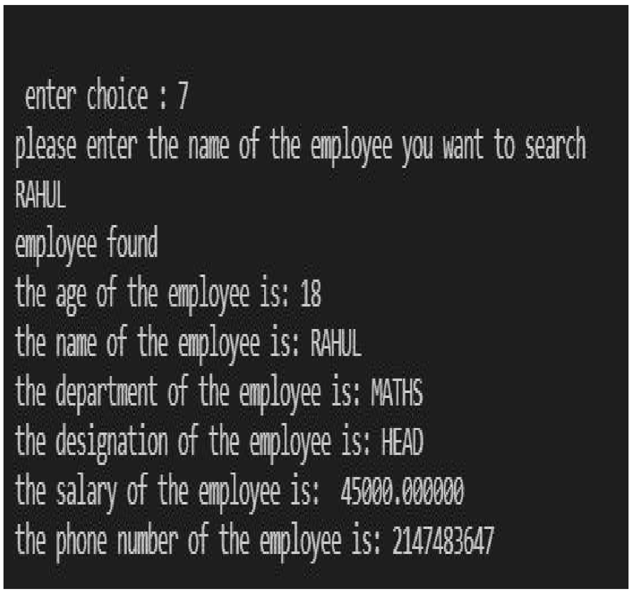
```
please enter choice from the following options

 1.Create a list of n number of employees
 2.Display from beginning
 3.Insert data at the end
 4.Delete data at the end
 5.Insert data at the beginning
 6.Delete data at the beginning
 7.Search an employee by name
 8.Delete an employee by name
 9.Exit
----------------------

 enter choice : 1

 enter number of employees:3

enter the details for employee 1

enter the age of the employee : 18
enter the name of the employee : PRASHANT
enter the department of the employee:CHEMISTRY
enter the designation of the employee : HEAD
enter the salary of the employee :  45000
enter the phone number of the employee : 9899768922

enter the details for employee 2

enter the age of the employee : 18
enter the name of the employee : PRATUESH
enter the department of the employee:PHYSICS
enter the designation of the employee : HEAD
enter the salary of the employee :  450000
enter the phone number of the employee : 8765678900

enter the details for employee 3

enter the age of the employee : 18
enter the name of the employee : RAHUL
enter the department of the employee:MATHS
enter the designation of the employee : HEAD
enter the salary of the employee :  45000
enter the phone number of the employee : 6756998702
```

- INSERTING AT END



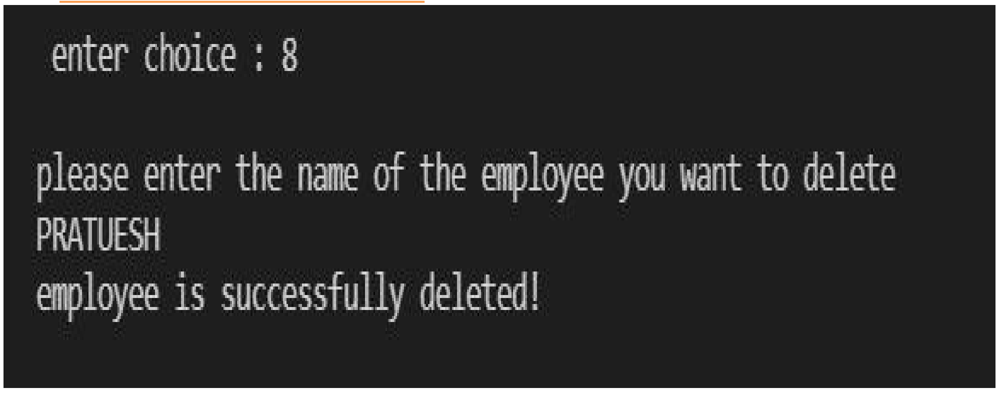
```
 enter choice : 3

enter the age of the employee : 22
enter the name of the employee : SHYAM
enter the department of the employee:COMPUTER
enter the designation of the employee : HEAD
enter the salary of the employee :  450000
enter the phone number of the employee : 9878654311
```

- SEARCHING BY NAME



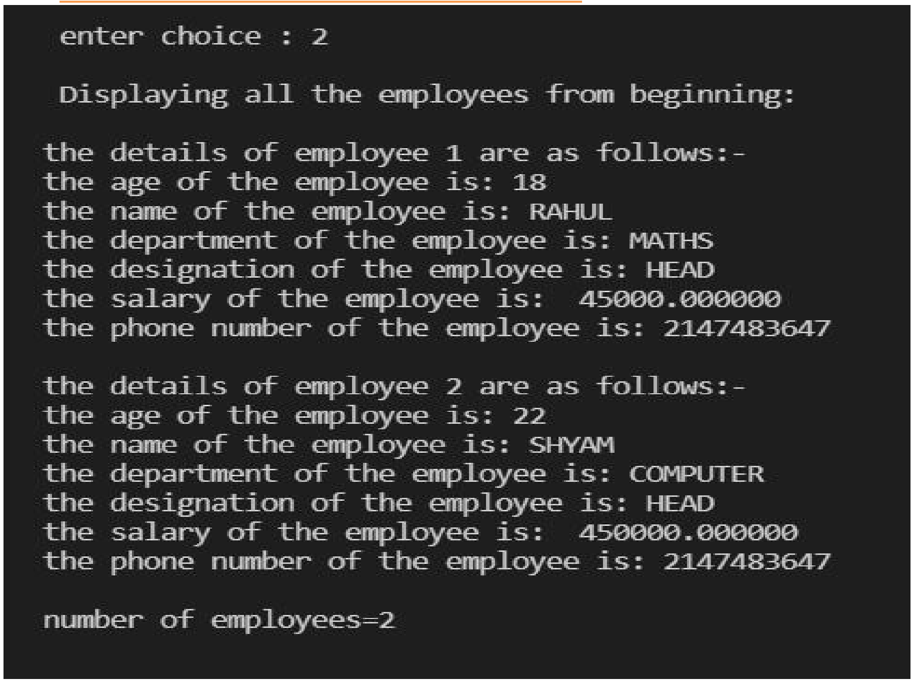
```
 enter choice : 7
please enter the name of the employee you want to search
RAHUL
employee found
the age of the employee is: 18
the name of the employee is: RAHUL
the department of the employee is: MATHS
the designation of the employee is: HEAD
the salary of the employee is:  45000.000000
the phone number of the employee is: 2147483647
```

- DELETING BY NAME



```
 enter choice : 8

please enter the name of the employee you want to delete
PRATUESH
employee is successfully deleted!
```

- DISPLAYING THE FINAL LIST



```
 enter choice : 2

 Displaying all the employees from beginning:

the details of employee 1 are as follows:-
the age of the employee is: 18
the name of the employee is: RAHUL
the department of the employee is: MATHS
the designation of the employee is: HEAD
the salary of the employee is:  45000.000000
the phone number of the employee is: 2147483647

the details of employee 2 are as follows:-
the age of the employee is: 22
the name of the employee is: SHYAM
the department of the employee is: COMPUTER
the designation of the employee is: HEAD
the salary of the employee is:  450000.000000
the phone number of the employee is: 2147483647

number of employees=2
```

CONCLUSION

The project Employee management System is successfully designed by using the doubly-linked list data structure. The management system created in the project allows the user to display each detail of all the employees. The program also allows the user to create and delete an employee record. The advantages of doubly linked list over linked list can be shown from the program where the navigation in both the directions in a list is possible (unlike single linked list). The program follows modular approach where different functions are defined for performing the specific tasks and hence, the program is user friendly as the flow of control of the program is easy to understand.

REFERENCES

- An Introduction to Data Structures with Applications –Jean Paul Tremblay & Pal G.Sorenson
- https://www.youtube.com/
- https://www.geeksforgeeks.org/
- https://www.learncpp.com/
- https://www.javatpoint.com/