DSA [20ES117] COURSE PROJECT REPORT

On

"HOTEL ACCOMMODATION SYSTEM"

Developed By:

H.T.NO	STUDENT NAME
2103A51049	Kandi Bala Sai Nithin

Under the Guidance of

Dr. A. SIVA KRISHNA REDDY

Associate Professor

Submitted to

Department of Computer Science and Artificial Intelligence

SR University

Ananthasagar(V), Hasanparthy(M), Hanamkonda(Dist.) – 506371

www.sru.edu.in

December 2022

Department of Computer Science and Artificial Intelligence

CERTIFICATE

This is to certify that the DSA course project report entitled **"HOTEL ACCOMMODATION SYSTEM"** is a record of bona fide work carried out by the student **"Kandi Bala Sai Nithin"** bearing roll number **"2103A51049"** of Computer Science and Artificial Intelligence department during the academic year 2022-23.

Supervisor

(Dr. A. Siva Krishna Reddy)

INDEX

Sl. No	Title	Page No.
1.	Idea	4
2.	Problem statement	5
3.	Module-wise description	6-7
4.	Usage of Tools	8
5.	Block Diagram	9
6.	Knowledge required	10
7.	Solution	11
8.	Data Structures	12-25
9.	Source code	26-51
10.	Output	52-56
11.	Conclusion	57
12.	Reference	58

IDEA

- My idea is to provide an application that helps hotels to keep track of their room bookings of different kinds of rooms like Standard Rooms, Deluxe Rooms, and Premium Rooms.
- The hotel staff can easily access information of any room when needed.
- The application also provides security. The system can be accessed only by entering correct username and password.
- It is implemented using the Data Structure – Singly Linked List.

PROBLEM STATEMENT:

Develop a C program to dynamically read and store details of bookings of a hotel's lodge. The application also provides the following mentioned functionalities:

1. Hotel Registration. (Only for the first time)

   The hotel is to be registered by entering the required details.

2. Login with Credentials.

   We can login only by entering correct username and password.

3. Menu selection for the following actions: allotting a room, vacating a room, searching a room, displaying all rooms, changing credentials, and logging out.

- Allotting a room further gives 3 options to select the kind of room from: Standard, Deluxe, and Premium rooms. The customer details are to be entered and the room will be booked.

- To vacate a room, room number should be entered and customer details are to be confirmed.

- To search a room, room number should be entered, then it shows the details of the customer who is staying in the room.

- Displaying all rooms shows the details of all rooms, i.e., it shows the details of the customer who is staying in the room.

- To change credentials, current username and password are to be entered. Then, new credentials may be updated.

- Logging out prompts you to login again.

And many more good functions for highly efficient system performance. Whenever any one of the above-mentioned menu functions are executed, the respective menu options of either the user or admin are displayed iteratively.

MODULES:

In this application many of the variables and structure are declared globally so that these variables and structure members can be accessed throughout the program at any function call. We can choose any function by using function calls which are declared in switch-case. In order to repeat the loop, control statement (while) is used with condition. The memory allocation in this program is done dynamically.

In this application, the following modules are used:

1. hotelRegistration()

In this module, the application focuses on collecting information about the structure of the hotel i.e., number of floors in hotel, number of rooms in each floor and information about the rooms is collected i.e., about the kind of room (standard, deluxe or premium). And, the user has to provide initial username and password which can be changed later.

2. allotRoom()

In this module, the user can select which kind of room is to be booked. After entering the choice, room number and customer details are to be entered. Then, the room is booked.

3. vacateRoom()

In this module, the application asks the user to enter room number that is to be vacated. It displays the details of the customer and asks for confirmation to vacate the room.

4. searchRoom()

In this module, the application allows the user to search for a customer's details by entering the room number.

5. displayBookedRooms()

This module displays all booked rooms' customer details.

6. credentials()

In this module the user can change the username and password after entering the current username and password.

Other modules:

These are the modules which are used inside the above modules. These are created for better readability of the code.

- displayStandardRooms(): Displays all standard room numbers
- displayDeluxeRooms(): Displays all deluxe room numbers
- displayPremiumRooms(): Displays all premium room numbers

- displayAllRooms(): Displays all room numbers
- displayVacantStandard(): Displays all standard room numbers that are vacant
- displayVacantDeluxe(): Displays all deluxe room numbers that are vacant
- displayVacantPremium(): Displays all premium room numbers that are vacant
- displayVacantRooms(): Displays all room numbers that are vacant
- displayOccupiedRooms(): Displays all room numbers that are occupied

USAGE OF TOOLS

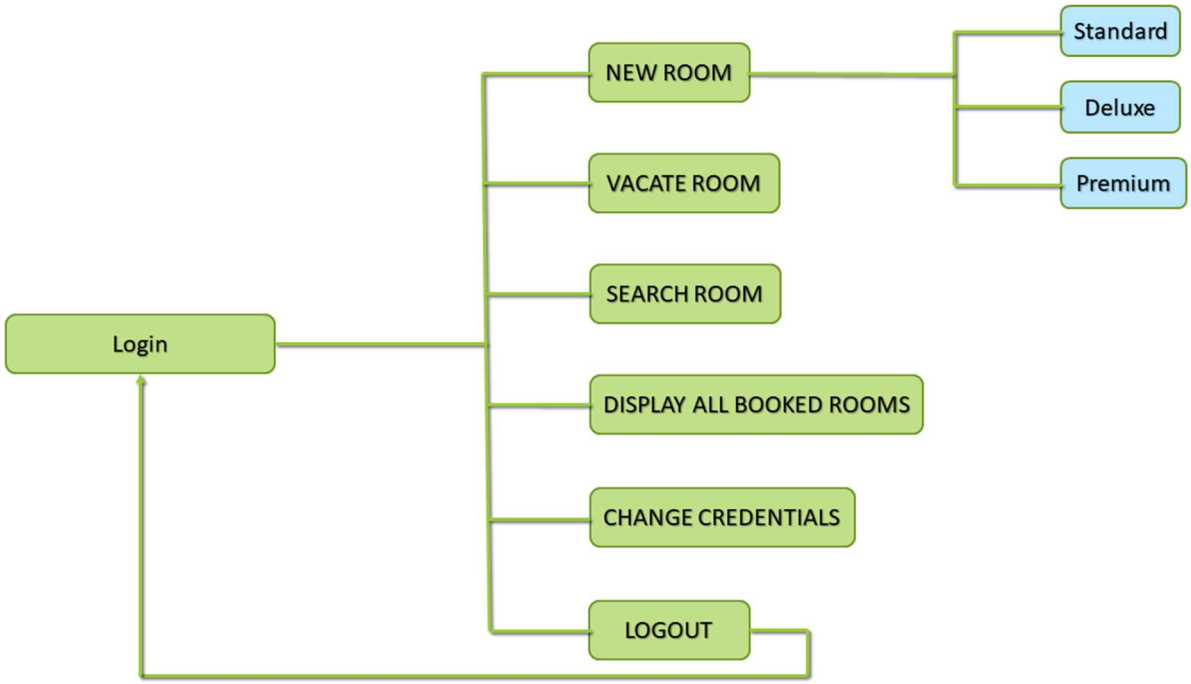
Hardware Tools:

- Operating System Windows 2000/NT/XP/Vista
- Ram 256 MB or more
- HARD DISK 40 GB or more
- Processor P3 or High

Software Tools:

Dev-C++ is a free full-featured integrated development environment distributed under the GNU General Public License for programming in C and C++. It is written in Delphi. It is bundled with, and uses, the MinGW or TDM-GCC 64bit port of the GCC as its compiler.

BLOCK DIAGRAM:



KNOWLEDGE REQUIRED TO DEVELOP THIS APPLICATION

- Control Statements (if, if-else, switch)
- Loop Statements
- Arrays
- Strings (strings) and its functions (strcmp)
- Functions (Any type of user defined functions)
- Structures
- Pointers (pointers to structures)
- Dynamic Memory Allocation (malloc/ calloc/ realloc)
- Linked Lists (Single Linked List)
- Windows library functions

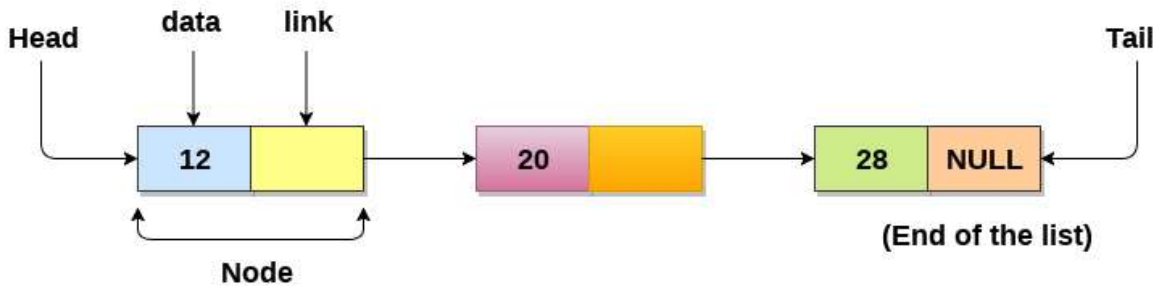
SOLUTION

- The application can be implemented in hotels in order to manage its customers and their information.
- Firstly, the interface takes the information of hotel and initial username and password.
- Then, it redirects to login page.
- After login, the application provides options to manage the bookings.
- This interface provides security as no one can open it without username and password.
- Here the data structure linked list can provide us the functionalities we require to make room bookings.

DATA STRUCTURES:

Single Linked List:

A linked list is a linear data structure that includes a series of connected nodes. Here, each node stores the data and the address of the next node. For example,



Linked List can be defined as collection of objects called nodes that are randomly stored in the memory.

A node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory.

The last node of the list contains pointer to the null.

Representation of Linked List:

Let's see how each node of the linked list is represented. Each node consists:

- A data item
- An address of another node

We wrap both the data item and the next node reference in a struct as:

```
struct node
{
int data;
struct node *next;
}
struct node *head=NULL;
```

Linked List Applications:

- Dynamic memory allocation
- Implemented in stack and queue
- In **undo** functionality of softwares
- Hash tables, Graphs

Uses of Linked Lists:

- The list is not required to be contiguously present in the memory. The node can reside anywhere in the memory and linked together to make a list. This achieves optimized utilization of space.
- List size is limited to the memory size and doesn't need to be declared in advance.
- Empty node cannot be present in the linked list.
- We can store values of primitive types or objects in the singly linked list.

Why use linked list over array?

Till now, we were using array data structure to organize the group of elements that are to be stored individually in the memory. However, Array has several advantages and disadvantages which must be known in order to decide the data structure which will be used throughout the program.

Array contains following limitations:

1. The size of array must be known in advance before using it in the program.
2. Increasing size of the array is a time taking process. It is almost impossible to expand the size of the array at run time.
3. All the elements in the array need to be contiguously stored in the memory. Inserting any element in the array needs shifting of all its predecessors.

Linked list is the data structure which can overcome all the limitations of an array. Using linked list is useful because,

1. It allocates the memory dynamically. All the nodes of linked list are non-contiguously stored in the memory and linked together with the help of pointers.

2. Sizing is no longer a problem since we do not need to define its size at declaration. List grows as per the program's demand and limited to the available memory space.

Linked List Complexity:

Time Complexity

	Worst case	Average Case
Search	O(n)	O(n)
Insert	O(1)	O(1)
Deletion	O(1)	O(1)

Space Complexity: O(n)

Linked List Operations:

Here's a list of basic linked list operations that we will cover in this article.

- Traversal - access each element of the linked list
- Insertion - adds a new element to the linked list
    - Insertion can be done at beginning, ending and after a specific node
- Deletion - removes the existing elements
    - Deletion can be done at the beginning , ending and after a specific node.

Traverse a Linked List:

Traversing is the most common operation that is performed in almost every scenario of singly linked list. Traversing means visiting each node of the list once in order to perform some operation on that. This will be done by using the following statements.

```
ptr = head;
      while (ptr!=NULL)
         {
            ptr = ptr -> next;
         }
```

Algorithm:

- **STEP 1:** SET PTR = HEAD
- **STEP 2:** IF PTR = NULL
  WRITE "EMPTY LIST"
  GOTO STEP 7
  END OF IF
- **STEP 4:** REPEAT STEP 5 AND 6 UNTIL PTR != NULL
- **STEP 5:** PRINT PTR→ DATA
- **STEP 6:** PTR = PTR → NEXT
  [END OF LOOP]
- **STEP 7:** EXIT

Insertion of Element in Single Linked List at Beginning:

Inserting a new element into a singly linked list at beginning is quite simple. We just need to make a few adjustments in the node links. There are the following steps which need to be followed in order to inser a new node in the list at beginning.

- Allocate the space for the new node and store data into the data part of the node. This will be done by the following statements.

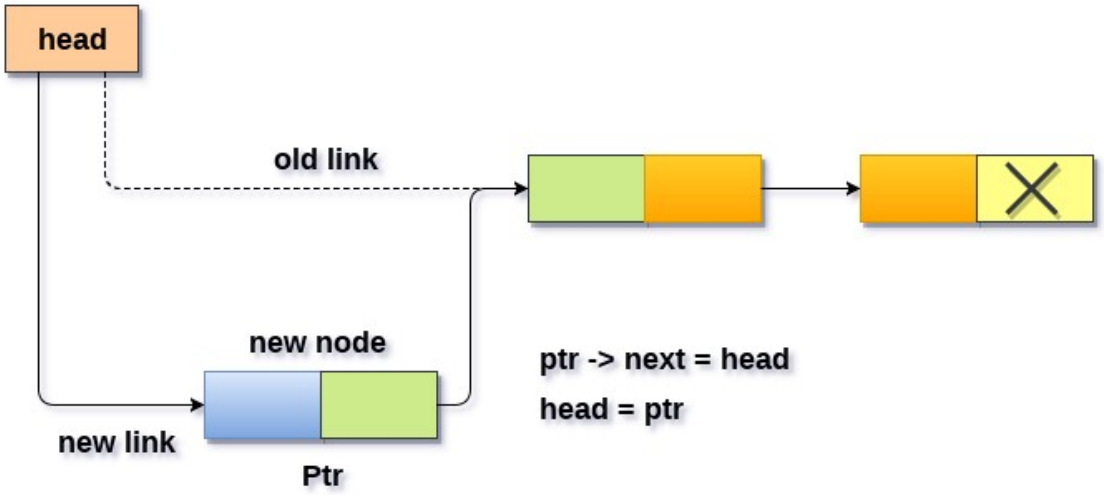
1. ptr = (struct node *) malloc(sizeof(struct node *));
2. ptr → data = item

- Make the link part of the new node pointing to the existing first node of the list. This will be done by using the following statement.

1. ptr->next = head;

- At the last, we need to make the new node as the first node of the list this will be done by using the following statement.

1. head = ptr;



Algorithm:

- **Step 1:** IF PTR = NULL

  Write OVERFLOW
  Go to Step 7
  [END OF IF]

- **Step 2:** SET NEW_NODE = PTR
- **Step 3:** SET PTR = PTR → NEXT
- **Step 4:** SET NEW_NODE → DATA = VAL
- **Step 5:** SET NEW_NODE → NEXT = HEAD
- **Step 6:** SET HEAD = NEW_NODE
- **Step 7:** EXIT

Insertion of Element in Single Linked List at the End:

In order to insert a node at the last, there are two following scenarios which need to be mentioned.

1. The node is being added to an empty list
2. The node is being added to the end of the linked list

in the first case,

- The condition (head == NULL) gets satisfied. Hence, we just need to allocate the space for the node by using malloc statement in C. Data and the link part of the node

are set up by using the following statements.

1. ptr->data = item;
2. ptr -> next = NULL;

- Since, **ptr** is the only node that will be inserted in the list hence, we need to make this node pointed by the head pointer of the list. This will be done by using the following Statements.

1. Head = ptr

In the second case,

- The condition **Head = NULL** would fail, since Head is not null. Now, we need to declare a temporary pointer temp in order to traverse through the list. **temp** is made to point the first node of the list.

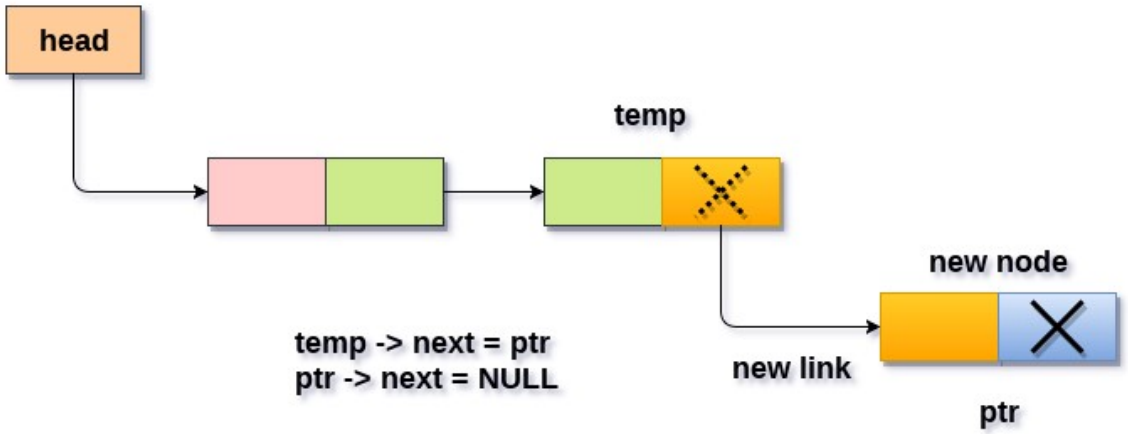
1. Temp = head

- Then, traverse through the entire linked list using the statements:

1. **while** (temp→ next != NULL)
2. temp = temp → next;

- At the end of the loop, the temp will be pointing to the last node of the list. Now, allocate the space for the new node, and assign the item to its data part. Since, the new node is going to be the last node of the list hence, the next part of this node needs to be pointing to the **null**. We need to make the next part of the temp node (which is currently the last node of the list) point to the new node (ptr) .

1. temp = head;
2. **while** (temp -> next != NULL)
3. {
4. temp = temp -> next;
5. }
6. temp->next = ptr;
7. ptr->next = NULL;



Algorithm:

- **Step 1:** IF PTR = NULL Write OVERFLOW
  Go to Step 1
  [END OF IF]
- **Step 2:** SET NEW_NODE = PTR
- **Step 3:** SET PTR = PTR - > NEXT
- **Step 4:** SET NEW_NODE - > DATA = VAL
- **Step 5:** SET NEW_NODE - > NEXT = NULL
- **Step 6:** SET PTR = HEAD
- **Step 7:** Repeat Step 8 while PTR - > NEXT != NULL
- **Step 8:** SET PTR = PTR - > NEXT
  [END OF LOOP]
- **Step 9:** SET PTR - > NEXT = NEW_NODE
- **Step 10:** EXIT

Insertion of Element in Single Linked List after Specified Node:

- In order to insert an element after the specified number of nodes into the linked list, we need to skip the desired number of elements in the list to move the pointer at the position after which the node will be inserted. This will be done by using the following statements.

1. temp=head;
2. **for**(i=0;i<loc;i++)

```
3.          {
4.              temp = temp->next;
5.              if(temp == NULL)
6.              {
7.                  return;
8.              }
9.
10.         }
```

- Allocate the space for the new node and add the item to the data part of it. This will be done by using the following statements.

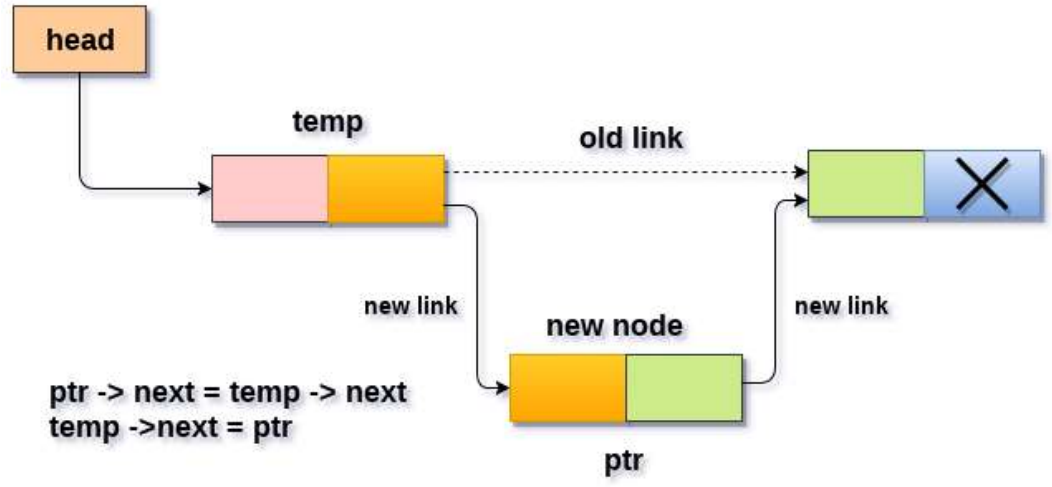
```
1. ptr = (struct node *) malloc (sizeof(struct node));
2.       ptr->data = item;
```

- Now, we just need to make a few more link adjustments and our node at will be inserted at the specified position. Since, at the end of the loop, the loop pointer temp would be pointing to the node after which the new node will be inserted. Therefore, the next part of the new node ptr must contain the address of the next part of the temp (since, ptr will be in between temp and the next of the temp). This will be done by using the following statements.

```
1. ptr→ next = temp → next
```

- Now, we just need to make the next part of the temp, point to the new node ptr. This will insert the new node ptr, at the specified position.

```
1. temp ->next = ptr;
```



Algorithm:

- **STEP 1:** IF PTR = NULL

WRITE OVERFLOW
GOTO STEP 12
END OF IF

- **STEP 2:** SET NEW_NODE = PTR
- **STEP 3:** NEW_NODE → DATA = VAL
- **STEP 4:** SET TEMP = HEAD
- **STEP 5:** SET I = 0
- **STEP 6:** REPEAT STEP 5 AND 6 UNTIL I<loc< li=""></loc<>
- **STEP 7:** TEMP = TEMP → NEXT
- **STEP 8:** IF TEMP = NULL

WRITE "DESIRED NODE NOT PRESENT"
GOTO STEP 12
END OF IF
END OF LOOP

- **STEP 9:** PTR → NEXT = TEMP → NEXT
- **STEP 10:** TEMP → NEXT = PTR
- **STEP 11:** SET PTR = NEW_NODE
- **STEP 12:** EXIT

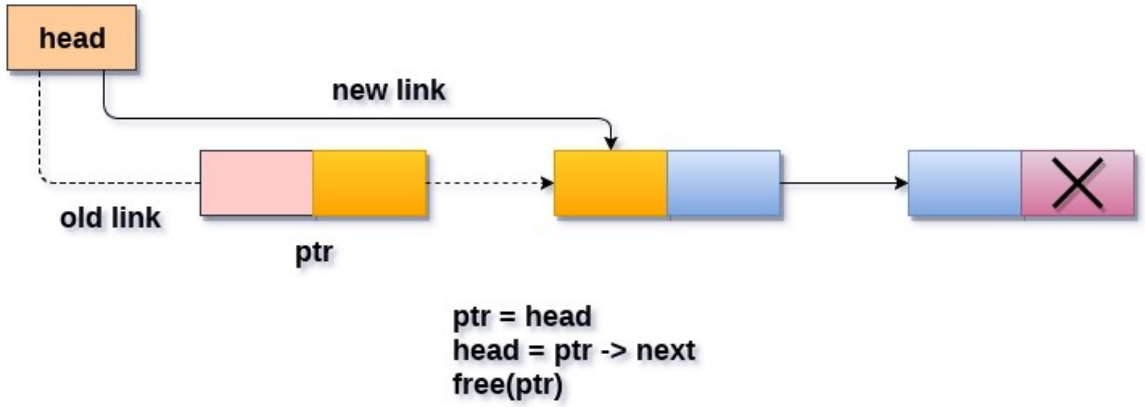
Deletion of Element in Single Linked List at Beginning:

Deleting a node from the beginning of the list is the simplest operation of all. It just need a few adjustments in the node pointers. Since the first node of the list is to be deleted, therefore, we just need to make the head, point to the next of the head. This will be done by using the following statements.

1. ptr = head;
2. head = ptr->next;

Now, free the pointer ptr which was pointing to the head node of the list. This will be done by using the following statement.

1. free(ptr)



Algorithm:

- **Step 1:** IF HEAD = NULL

Write UNDERFLOW
Go to Step 5
[END OF IF]

- **Step 2:** SET PTR = HEAD
- **Step 3:** SET HEAD = HEAD -> NEXT
- **Step 4:** FREE PTR
- **Step 5:** EXIT

Deletion of Element in Single Linked List at the End:

There are two scenarios in which, a node is deleted from the end of the linked list.

1. There is only one node in the list and that needs to be deleted.
2. There are more than one node in the list and the last node of the list will be deleted.

In the first scenario,

the condition head → next = NULL will survive and therefore, the only node head of the list will be assigned to null. This will be done by using the following statements.

1. ptr = head
2. head = NULL

3. free(ptr)

In the second scenario,

The condition head → next = NULL would fail and therefore, we have to traverse the node in order to reach the last node of the list.

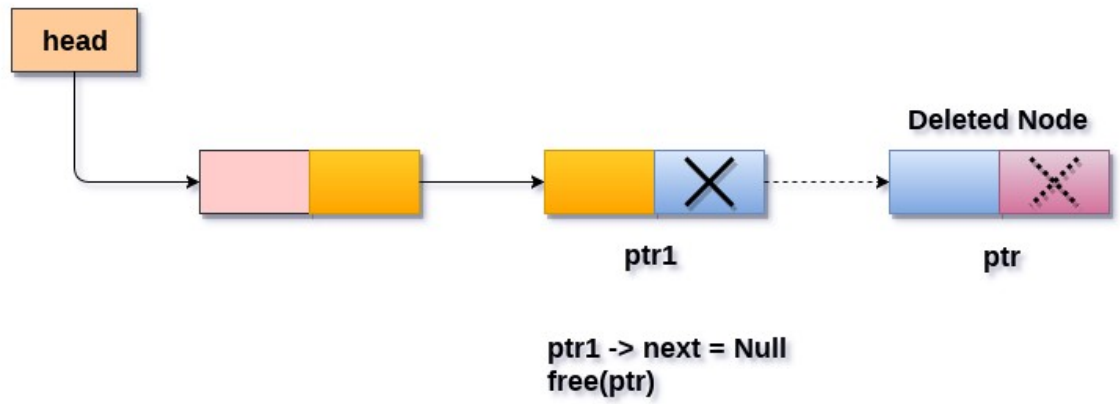
For this purpose, just declare a temporary pointer temp and assign it to head of the list. We also need to keep track of the second last node of the list. For this purpose, two pointers ptr and ptr1 will be used where ptr will point to the last node and ptr1 will point to the second last node of the list.

this all will be done by using the following statements.

```
1. ptr = head;
2.          while(ptr->next != NULL)
3.          {
4.             ptr1 = ptr;
5.             ptr = ptr ->next;
6.          }
```

Now, we just need to make the pointer ptr1 point to the NULL and the last node of the list that is pointed by ptr will become free. It will be done by using the following statements.

```
1. ptr1->next = NULL;
2. free(ptr);
```



Algorithm:

- **Step 1:** IF HEAD = NULL

Write UNDERFLOW
Go to Step 8
[END OF IF]

- **Step 2:** SET PTR = HEAD
- **Step 3:** Repeat Steps 4 and 5 while PTR -> NEXT!= NULL
- **Step 4:** SET PREPTR = PTR
- **Step 5:** SET PTR = PTR -> NEXT

[END OF LOOP]

- **Step 6:** SET PREPTR -> NEXT = NULL
- **Step 7:** FREE PTR
- **Step 8:** EXIT

Deletion of Element in Single Linked List after the Specified Node:

In order to delete the node, which is present after the specified node, we need to skip the desired number of nodes to reach the node after which the node will be deleted. We need to keep track of the two nodes. The one which is to be deleted the other one if the node which is present before that node. For this purpose, two pointers are used: ptr and ptr1.

Use the following statements to do so.

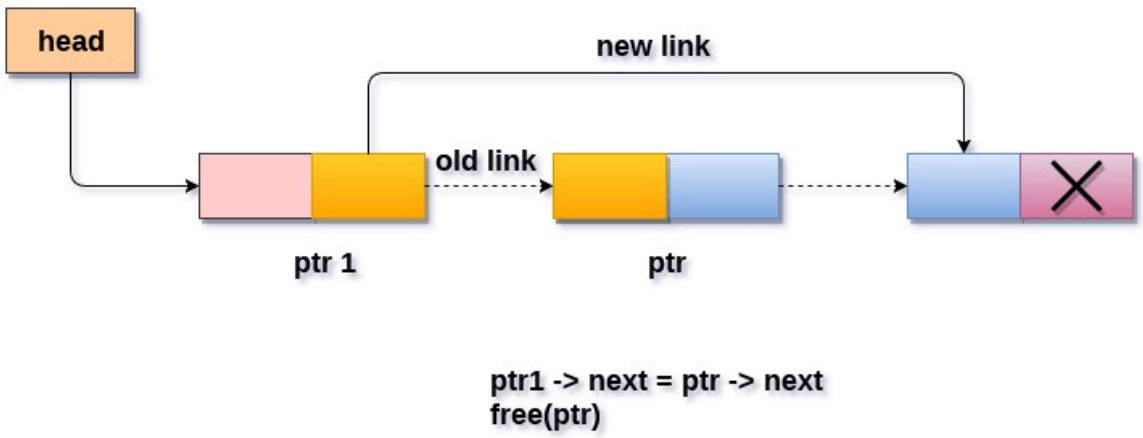
```
1. ptr=head;
2.         for(i=0;i<loc;i++)
3.         {
4.             ptr1 = ptr;
5.             ptr = ptr->next;
6.
7.             if(ptr == NULL)
8.             {
9.                 printf("\nThere are less than %d elements in the list..",loc);
```

```
10.             return;
11.         }
12.     }
```

Now, our task is almost done, we just need to make a few pointer adjustments. Make the next of ptr1 (points to the specified node) point to the next of ptr (the node which is to be deleted).

This will be done by using the following statements.

```
1. ptr1 ->next = ptr ->next;
2.         free(ptr);
```



Algorithm:

- **STEP 1:** IF HEAD = NULL

WRITE UNDERFLOW
GOTO STEP 10
END OF IF

- **STEP 2:** SET TEMP = HEAD
- **STEP 3:** SET I = 0
- **STEP 4:** REPEAT STEP 5 TO 8 UNTIL I<loc< li=""></loc<>
- **STEP 5:** TEMP1 = TEMP
- **STEP 6:** TEMP = TEMP → NEXT
- **STEP 7:** IF TEMP = NULL

WRITE "DESIRED NODE NOT PRESENT"
GOTO STEP 12
END OF IF

- **STEP 8:** I = I+1

END OF LOOP

- **STEP 9:** TEMP1 → NEXT = TEMP → NEXT
- **STEP 10:** FREE TEMP
- **STEP 11:** EXIT

SOURCE CODE:

```
/* Dev: Nithin Kandi */
#include<stdio.h>
#include<stdlib.h>
#include<windows.h>
#include<string.h>
int rooms;
char username[10], password[10];
struct room
{
	int occupied;
	char roomType;
	int roomNumber;
	char name[20];
	long long int mobile,aadhaar;
	struct room *nextRoom;
};
struct room *first=NULL;
void displayStandardRooms();
void displayDeluxeRooms();
void displayPremiumRooms();
void displayAllRooms();
void displayVacantStandard();
void displayVacantDeluxe();
void displayVacantPremium();
void displayVacantRooms();
```

```
void displayOccupiedRooms();
void hotelRegistration();
void allotRoom();
void vacateRoom();
void searchRoom();
void displayBookedRooms();
void credentials();
int main()
{
	int choice,yes3=1,yes4=1;
	char username2[10],password2[10];
	hotelRegistration();
	displayAllRooms();
	printf("\nYOU WILL BE REDIRECTED TO THE MAIN PAGE IN 10 SECONDS");
	sleep(10);
	login:
	system("cls");
	printf("\n\n\n\n	Enter username: ");
	scanf("%s",username2);
	printf("\n	Enter password: ");
	scanf("%s",password2);
	yes3=strcmp(username,username2);
	yes4=strcmp(password,password2);
	if((yes3)||(yes4))
	{
		sleep(1);
		printf("\n	INVALID CREDENTIALS\n");
```

```
            printf("       PLEASE TRY AGAIN\n");
            sleep(3);
            goto login;
    }
    else
    {
            while(1)
            {
                    system("cls");
                    printf("\n    1 Allot a room\n");
                    printf("    2 Vacate a room\n");
                    printf("    3 Search a customer's details\n");
                    printf("    4 Display all booked room details\n");
                    printf("    5 Change credentials\n");
                    printf("    6 Logout\n");
                    printf("\n    Enter your choice: ");
                    scanf("%d",&choice);
                    switch(choice)
                    {
                            case 1:
                                    system("cls");
                                    allotRoom();
                                    break;
                            case 2:
                                    system("cls");
                                    vacateRoom();
                                    break;
```

```
                    case 3:
                        system("cls");
                        searchRoom();
                        break;
                    case 4:
                        system("cls");
                        displayBookedRooms();
                        break;
                    case 5:
                        credentials();
                        break;
                    case 6:
                        goto login;
                        break;
                    default:
                        printf("\n    ENTER VALID OPTION");
                        sleep(5);
                        break;
                }
            }
        }
        return 0;
}
void hotelRegistration()
{
        printf("SET USERNAME: ");
        scanf("%s",username);
```

```
printf("SET PASSWORD: ");
scanf("%s",password);
int floors;
int i,j,k,l;
int std,del,prem;
int standard[100],deluxe[100],premium[100];
printf("\n\nEnter number of floors: ");
scanf("%d",&floors);
printf("Enter number of rooms in each floor: ");
scanf("%d",&rooms);
struct room *tempRoom,*newRoom;
tempRoom=(struct room*)malloc(sizeof(struct room));
tempRoom->roomNumber=101;
tempRoom->occupied=0;
tempRoom->nextRoom=NULL;
first=tempRoom;
for(i=1;i<=floors;i++)
{
        for(j=1;j<=rooms;j++)
        {
                if(i==1&&j==1)
                {
                        continue;
                }
                newRoom=(struct room*)malloc(sizeof(struct room));
                newRoom->roomNumber=i*100+j;
                newRoom->occupied=0;
```

```
                newRoom->nextRoom=NULL;
                tempRoom->nextRoom=newRoom;
                tempRoom=newRoom;
        }
}
printf("\nAll room numbers are:\n");
l=0;
tempRoom=first;
while(tempRoom!=NULL)
{
        l++;
        printf("%d\t",tempRoom->roomNumber);
        tempRoom=tempRoom->nextRoom;
        if(l==rooms)
        {
                printf("\n");
                l=0;
        }
}
roomTypes:{
printf("\nEnter number of standard rooms: ");
scanf("%d",&std);
printf("Enter number of deluxe rooms: ");
scanf("%d",&del);
printf("Enter number of premium rooms: ");
scanf("%d",&prem);
if((std+del+prem)==(floors*rooms))
```

```
{
printf("\nEnter room numbers of all standard rooms:\n");
for(k=0;k<std;k++)
{
        scanf("%d",&standard[k]);
}
printf("Enter room numbers of all deluxe rooms:\n");
for(k=0;k<del;k++)
{
        scanf("%d",&deluxe[k]);
}
printf("Enter room numbers of all premium rooms:\n");
for(k=0;k<prem;k++)
{
        scanf("%d",&premium[k]);
}
for(k=0;k<std;k++)
{
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                if(standard[k]==tempRoom->roomNumber)
                {
                        tempRoom->roomType='S';
                        break;
                }
                tempRoom=tempRoom->nextRoom;
```

```
            }
        }
        for(k=0;k<del;k++)
        {
            tempRoom=first;
            while(tempRoom!=NULL)
            {
                if(deluxe[k]==tempRoom->roomNumber)
                {
                    tempRoom->roomType='D';
                    break;
                }
                tempRoom=tempRoom->nextRoom;
            }
        }
        for(k=0;k<prem;k++)
        {
            tempRoom=first;
            while(tempRoom!=NULL)
            {
                if(premium[k]==tempRoom->roomNumber)
                {
                    tempRoom->roomType='P';
                    break;
                }
                tempRoom=tempRoom->nextRoom;
            }
```

```
		}
		}
		else
		{
			printf("Total number of rooms not equal to entered number of rooms\n");
			printf("PLEASE TRY AGAIN\n");
			goto roomTypes;
		}
		}
		system("cls");
		for(k=0;k<3;k++)
		{
			printf("\nYou will be redirected to the room numbers page in %d seconds\n",3-k);
			sleep(1);
			system("cls");
		}
}
void displayStandardRooms()
{
		int l;
		struct room *tempRoom;
		printf("\nAll standard rooms are:\n");
		l=0;
		tempRoom=first;
		while(tempRoom!=NULL)
		{
```

```
                l++;
                if(tempRoom->roomType=='S')
                {
                        printf("%d\t",tempRoom->roomNumber);
                }
                tempRoom=tempRoom->nextRoom;
                if(l==rooms)
                {
                        printf("\n");
                        l=0;
                }
        }
}
void displayDeluxeRooms()
{
        int l;
        struct room *tempRoom;
        printf("\nAll deluxe rooms are:\n");
        l=0;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                l++;
                if(tempRoom->roomType=='D')
                {
                        printf("%d\t",tempRoom->roomNumber);
                }
```

```
                tempRoom=tempRoom->nextRoom;
                if(l==rooms)
                {
                        printf("\n");
                        l=0;
                }
        }
}
void displayPremiumRooms()
{
        int l;
        struct room *tempRoom;
        printf("\nAll premium rooms are:\n");
        l=0;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                l++;
                if(tempRoom->roomType=='P')
                {
                        printf("%d\t",tempRoom->roomNumber);
                }
                tempRoom=tempRoom->nextRoom;
                if(l==rooms)
                {
                        printf("\n");
                        l=0;
```

```
                }
        }
}
void displayAllRooms()
{
        int l;
        struct room *tempRoom;
        printf("\nAll room numbers are:\n");
        l=0;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                l++;
                printf("%d\t",tempRoom->roomNumber);
                tempRoom=tempRoom->nextRoom;
                if(l==rooms)
                {
                        printf("\n");
                        l=0;
                }
        }
        displayStandardRooms();
        displayDeluxeRooms();
        displayPremiumRooms();
}
void displayVacantStandard()
{
```

```
        int l;
        struct room *tempRoom;
        printf("\nAll available standard rooms are:\n");
        l=0;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                l++;
                if(tempRoom->roomType=='S'&&tempRoom->occupied==0)
                {
                        printf("%d\t",tempRoom->roomNumber);
                }
                tempRoom=tempRoom->nextRoom;
                if(l==rooms)
                {
                        printf("\n");
                        l=0;
                }
        }
}
void displayVacantDeluxe()
{
        int l;
        struct room *tempRoom;
        printf("\nAll available deluxe rooms are:\n");
        l=0;
        tempRoom=first;
```

```
        while(tempRoom!=NULL)
        {
                l++;
                if(tempRoom->roomType=='D'&&tempRoom->occupied==0)
                {
                        printf("%d\t",tempRoom->roomNumber);
                }
                tempRoom=tempRoom->nextRoom;
                if(l==rooms)
                {
                        printf("\n");
                        l=0;
                }
        }
}
void displayVacantPremium()
{
        int l;
        struct room *tempRoom;
        printf("\nAll available premium rooms are:\n");
        l=0;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                l++;
                if(tempRoom->roomType=='P'&&tempRoom->occupied==0)
                {
```

```
                printf("%d\t",tempRoom->roomNumber);
            }
            tempRoom=tempRoom->nextRoom;
            if(l==rooms)
            {
                printf("\n");
                l=0;
            }
        }
}
void displayVacantRooms()
{
        int l;
        struct room *tempRoom;
        printf("\nAll vacant rooms are:\n");
        l=0;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
            l++;
            if(tempRoom->occupied==0)
            {
                printf("%d\t",tempRoom->roomNumber);
            }
            tempRoom=tempRoom->nextRoom;
            if(l==rooms)
            {
```

```
                printf("\n");
                l=0;
            }
        }
        displayVacantStandard();
        displayVacantDeluxe();
        displayVacantPremium();
}
void displayOccupiedRooms()
{
        int l;
        struct room *tempRoom;
        printf("\nAll Occupied rooms are:\n");
        l=0;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
            l++;
            if(tempRoom->occupied==1)
            {
                printf("%d\t",tempRoom->roomNumber);
            }
            tempRoom=tempRoom->nextRoom;
            if(l==rooms)
            {
                printf("\n");
                l=0;
```

```
                }
        }
}
void allotRoom()
{
        int choice,roomno=0;
        struct room *tempRoom;
        tempRoom=first;
        printf("What kind of room do you want?\n");
        printf("1 Standard\n");
        printf("2 Deluxe\n");
        printf("3 Premium\n");
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
                case 1:
                        displayVacantStandard();
                        printf("Enter room number from any of the above rooms: ");
                        scanf("%d",&roomno);
                        break;
                case 2:
                        displayVacantDeluxe();
                        printf("Enter room number from any of the above rooms: ");
                        scanf("%d",&roomno);
                        break;
                case 3:
```

```
                displayVacantPremium();
                printf("Enter room number from any of the above rooms: ");
                scanf("%d",&roomno);
                break;
        default:
                sleep(1);
                printf("INVALID OPTION\n");
                sleep(2);
                break;
}
if(roomno!=0)
{
        while(tempRoom!=NULL)
        {
                if(tempRoom->roomNumber==roomno)
                {
                        break;
                }
                tempRoom=tempRoom->nextRoom;
        }
        if(tempRoom==NULL)
        {
                sleep(1);
                printf("PLEASE ENTER VALID ROOM NUMBER\n");
                sleep(3);
        }
        else
```

```
                {
                        printf("Enter customer details\n");
                        printf("Enter customer name: ");
                        scanf("%s",tempRoom->name);
                        printf("Enter customer mobile number: ");
                        scanf("%lld",&tempRoom->mobile);
                        printf("Enter customer aadhaar number: ");
                        scanf("%lld",&tempRoom->aadhaar);
                        tempRoom->occupied=1;
                        sleep(1);
                        printf("\nROOM ALLOTED!");
                        sleep(2);
                }
        }
}
void vacateRoom()
{
        int roomno,yes,yes1=0;
        struct room *tempRoom;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                if(tempRoom->occupied==1)
                {
                        yes1=1;
                }
                tempRoom=tempRoom->nextRoom;
```

```
}
if(yes1==1)
{
        displayOccupiedRooms();
        printf("Enter room number to vacate: ");
        scanf("%d",&roomno);
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                if(tempRoom->roomNumber==roomno)
                {
                        break;
                }
                tempRoom=tempRoom->nextRoom;
        }
        if(tempRoom==NULL)
        {
                sleep(1);
                printf("PLEASE ENTER VALID ROOM NUMBER\n");
                sleep(3);
        }
        else if(tempRoom->occupied==0)
        {
                sleep(1);
                printf("Room is already Vacant\n");
                sleep(1);
                printf("ONLY BOOKED ROOMS CAN BE VACATED\n");
```

```
                sleep(3);
        }
        else
        {
                printf("Customer name: %s\n",tempRoom->name);
                printf("Customer mobile number: %lld\n",tempRoom->mobile);
                printf("Customer aadhaar number: %lld\n",tempRoom->aadhaar);
                printf("Enter 1 to confirm\n");
                scanf("%d",&yes);
                if(yes==1)
                {
                        tempRoom->occupied=0;
                        sleep(1);
                        printf("\nROOM VACATED!");
                        sleep(3);
                }
                else
                {
                        sleep(1);
                        printf("\nPLEASE TRY AGAIN\n");
                        sleep(3);
                }
        }
}
else
{
        sleep(1);
```

```
                printf("NO ROOMS BOOKED TO VACATE");
                sleep(3);
        }
}
void searchRoom()
{
        int roomno,yes,yes1=0;
        struct room *tempRoom;
        tempRoom=first;
        while(tempRoom!=NULL)
        {
                if(tempRoom->occupied==1)
                {
                        yes1=1;
                }
                tempRoom=tempRoom->nextRoom;
        }
        if(yes1==1)
        {
                displayOccupiedRooms();
                printf("Enter room number: ");
                scanf("%d",&roomno);
                tempRoom=first;
                while(tempRoom!=NULL)
                {
                        if(tempRoom->roomNumber==roomno)
                        {
```

```
                    break;
                }
                tempRoom=tempRoom->nextRoom;
            }
            if(tempRoom==NULL)
            {
                sleep(1);
                printf("\nPLEASE ENTER VALID ROOM NUMBER\n");
                sleep(3);
            }
            else
            {
                if(tempRoom->occupied==0)
                {
                    sleep(1);
                    printf("\nRoom is Vacant\n");
                    sleep(1);
                    printf("PLEASE SELECT ONLY BOOKED ROOMS");
                    sleep(3);
                }
                else
                {
                    printf("Customer name: %s\n",tempRoom->name);
                    printf("Customer mobile number: %lld\n",tempRoom-
>mobile);
                    printf("Customer aadhaar number: %lld\n",tempRoom-
>aadhaar);
                    printf("Enter 1 to continue\n");
```

```
                        scanf("%d",&yes);
                    }
                }
            }
            else
            {
                sleep(1);
                printf("NO ROOMS BOOKED TO SEARCH");
                sleep(3);
            }
}
void displayBookedRooms()
{
    int roomno,yes,yes1=0;
    struct room *tempRoom;
    tempRoom=first;
    while(tempRoom!=NULL)
    {
        if(tempRoom->occupied==1)
        {
            yes1=1;
        }
        tempRoom=tempRoom->nextRoom;
    }
    tempRoom=first;
    if(yes1==1)
    {
```

```
            printf("Room\tName\tMobile\t\tAadhaar\n");
            while(tempRoom!=NULL)
            {
                if(tempRoom->occupied==1)
                {
                    printf("%d\t",tempRoom->roomNumber);
                    printf("%s\t",tempRoom->name);
                    printf("%lld\t",tempRoom->mobile);
                    printf("%lld\n",tempRoom->aadhaar);
                }
                tempRoom=tempRoom->nextRoom;
            }
            printf("Enter 1 to continue\n");
            scanf("%d",&yes);
        }
        else
        {
            printf("NO ROOMS BOOKED TO DISPLAY\n");
            printf("Enter 1 to continue\n");
            scanf("%d",&yes);
        }
}
void credentials()
{
        system("cls");
        int yes1=1,yes2=1;
        char username1[10],password1[10];
```

```
        printf("\nEnter current username: ");
        scanf("%s",username1);
        printf("Enter current password: ");
        scanf("%s",password1);
        yes1=strcmp(username,username1);
        yes2=strcmp(password,password1);
        if((yes1)||(yes2))
        {
                sleep(1);
                printf("INVALID CREDENTIALS\n");
                sleep(1);
                printf("PLEASE TRY AGAIN\n");
                sleep(3);
        }
        else
        {
                printf("\nEnter new username: ");
                scanf("%s",username);
                printf("Enter new password: ");
                scanf("%s",password);
                sleep(1);
                printf("\nCREDENTIALS CHANGED!");
                sleep(3);
        }
}
```

OUTPUT:

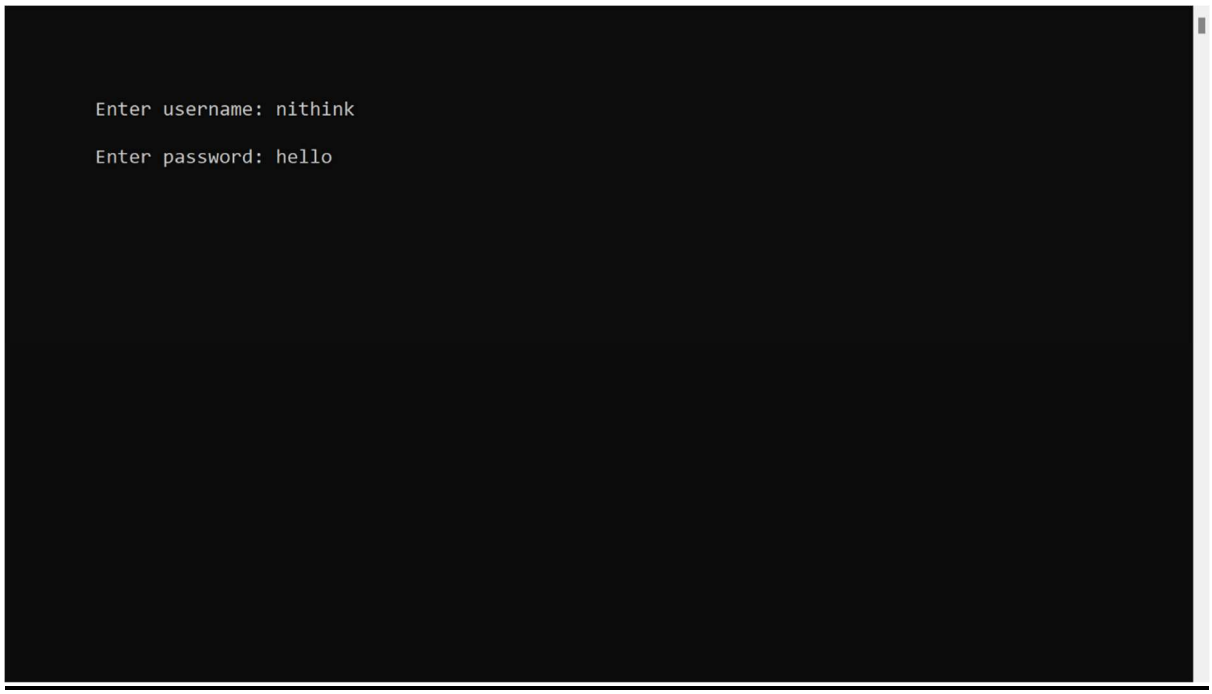
Hotel Registration

```
SET USERNAME: nithink
SET PASSWORD: hello


Enter number of floors: 2
Enter number of rooms in each floor: 4

All room numbers are:
101     102     103     104
201     202     203     204

Enter number of standard rooms: 4
Enter number of deluxe rooms: 2
Enter number of premium rooms: 2

Enter room numbers of all standard rooms:
101 102 103 104
Enter room numbers of all deluxe rooms:
201 202
Enter room numbers of all premium rooms:
203 204
```

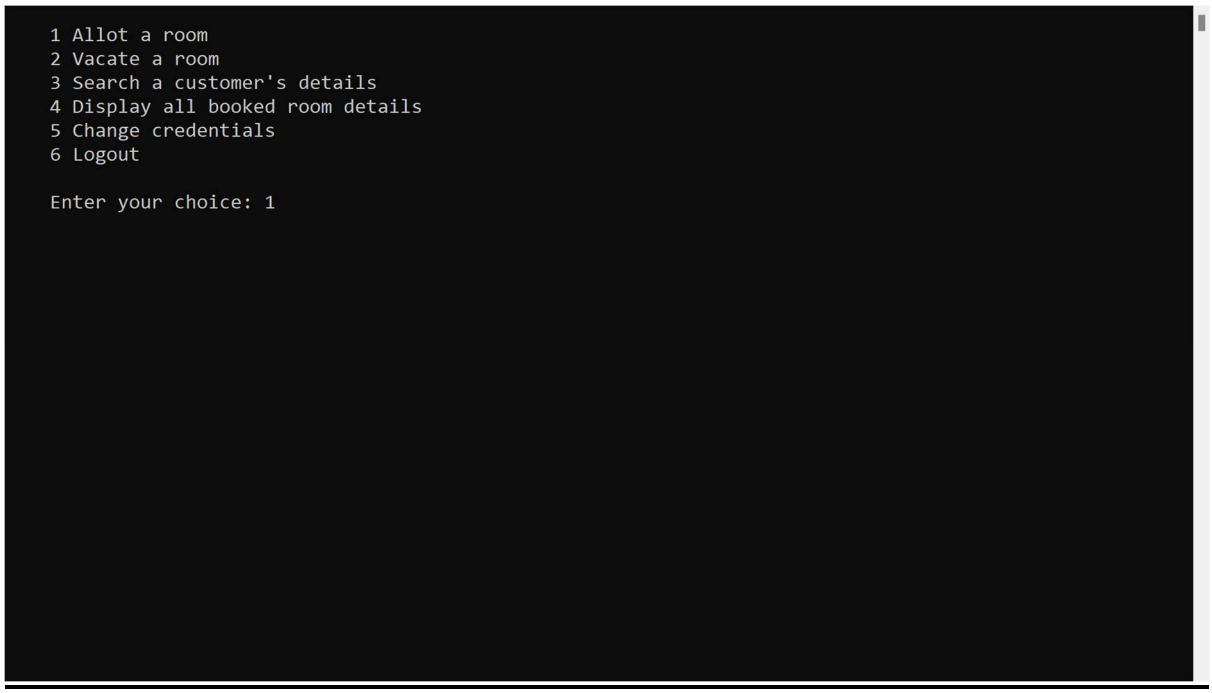
Brief display of room numbers

```
All room numbers are:
101     102     103     104
201     202     203     204

All standard rooms are:
101     102     103     104


All deluxe rooms are:

201     202

All premium rooms are:

203     204

YOU WILL BE REDIRECTED TO THE MAIN PAGE IN 10 SECONDS
```

Login



Main Menu



Allot room

```
What kind of room do you want?
1 Standard
2 Deluxe
3 Premium

Enter your choice: 3

All available premium rooms are:

203     204
Enter room number from any of the above rooms: 203
Enter customer details
Enter customer name: Nithin
Enter customer mobile number: 8639749574
Enter customer aadhaar number: 999988887777

ROOM ALLOTED!
```

Vacate room

```
All Occupied rooms are:

203
Enter room number to vacate: 203
Customer name: Nithin
Customer mobile number: 8639749574
Customer aadhaar number: 999988887777
Enter 1 to confirm
1

ROOM VACATED!
```

Search room

```
All Occupied rooms are:

203
Enter room number: 203
Customer name: Nithin
Customer mobile number: 8639749574
Customer aadhaar number: 999988887777
Enter 1 to continue
```

Display rooms

```
Room    Name    Mobile          Aadhaar
203     Nithin  8639749574      999988887777
Enter 1 to continue
```

Change Credentials

```
Enter current username: nithink
Enter current password: hello

Enter new username: hello
Enter new password: hello1

CREDENTIALS CHANGED!
```

Conclusion

From this project I want to conclude that the knowledge we gained by learning the Data Structures Laboratory has been applied successfully and the project can be implemented in any system that meet the specified requirements.

- This application can also be extended to implement in other environments like Library, Internet Cafes etc.
- In future changes can be done by linking the admin's mobile number for more security.

References

- Data Structures (Linked List) in Geeks for Geeks.
- Problems in most of the Hotels (Hard way