



main.c

Output



```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  typedef struct Student {
6      char studentID[20];    // now string,
7                              not int
8      char name[50];
9      int roomNo;
10     char block[10];
11     struct Student* next;
12 } Student;
13
14 Student* head = NULL;
15
16 // Function to create new node
17 Student* createNode(char id[], char name[]
18                     , int room, char block[]) {
19     Student* newNode = (Student*)malloc
20         (sizeof(Student));
21     strcpy(newNode->studentID, id);
22     strcpy(newNode->name, name);
23     newNode->roomNo = room;
24     strcpy(newNode->block, block);
25     newNode->next = NULL;
26     return newNode;
27 }
28
29 // Add new allotment
30 void addAllotment(char id[], char name[],
31                  int room, char block[]) {
```




main.c

Output



```
28     Student* newNode = createNode(id, name
    , room, block);
29     if (head == NULL) {
30         head = newNode;
31     } else {
32         Student* temp = head;
33         while (temp->next != NULL)
34             temp = temp->next;
35         temp->next = newNode;
36     }
37     printf("Allotment added
    successfully!\n");
38 }
39
40 // Remove student by ID
41 void removeStudent(char id[]) {
42     if (head == NULL) {
43         printf("No records found.\n");
44         return;
45     }
46     Student* temp = head;
47     Student* prev = NULL;
48     while (temp != NULL && strcmp(temp
    ->studentID, id) != 0) {
49         prev = temp;
50         temp = temp->next;
51     }
52     if (temp == NULL) {
53         printf("Student ID %s not found
    .\n", id);
54         return;
55     }
```




main.c

Output



```
56 ▾    if (prev == NULL) {
57        head = head->next;
58 ▾    } else {
59        prev->next = temp->next;
60    }
61    free(temp);
62    printf("Student with ID %s removed
        successfully.\n", id);
63 }
64
65 // Search by name
66 ▾ void searchByName(char name[]) {
67     Student* temp = head;
68     int found = 0;
69 ▾    while (temp != NULL) {
70 ▾        if (strcmp(temp->name, name) == 0)
71 ▾        {
72            printf("Found: ID=%s, Name=%s,
                Room=%d, Block=%s\n",
                temp->studentID, temp
                    ->name, temp->roomNo,
                    temp->block);
73            found = 1;
74        }
75        temp = temp->next;
```




main.c

Output



```
75         temp = temp->next;
76     }
77     if (!found) printf("No student with
                        name %s found.\n", name);
78 }
79
80 // Search by room number
81 void searchByRoom(int room) {
82     Student* temp = head;
83     int found = 0;
84     while (temp != NULL) {
85         if (temp->roomNo == room) {
86             printf("Found: ID=%s, Name=%s,
                        Room=%d, Block=%s\n",
87                     temp->studentID, temp
                        ->name, temp->roomNo,
                        temp->block);
88             found = 1;
89         }
90         temp = temp->next;
91     }
92     if (!found) printf("No student found
                        in room %d.\n", room);
93 }
94
95 // Display block-wise
96 void displayBlockWise(char block[]) {
97     Student* temp = head;
98     int found = 0;
99     printf("Students in Block %s:\n",
            block);
```




main.c

Output



```
100 while (temp != NULL) {
101     if (strcmp(temp->block, block) ==
        0) {
102         printf("ID=%s, Name=%s, Room
            =%d\n",
103             temp->studentID, temp
                ->name, temp->roomNo
                    );
104         found = 1;
105     }
106     temp = temp->next;
107 }
108 if (!found) printf("No students in
    block %s.\n", block);
109 }
110
111 // Reverse display using recursion
112 void reverseDisplay(Student* node) {
113     if (node == NULL) return;
114     reverseDisplay(node->next);
115     printf("ID=%s, Name=%s, Room=%d, Block
        =%s\n",
116         node->studentID, node->name,
            node->roomNo, node->block);
117 }
138 // Count students per block
139 void countPerBlock() {
140     Student* temp = head;
141     int countA = 0, countB = 0, countC = 0
        , countOther = 0;
142     while (temp != NULL) {
143         if (strcmp(temp->block, "A") == 0)
            countA++;
144         else if (strcmp(temp->block, "B")
            == 0) countB++;
145         else if (strcmp(temp->block, "C")
            == 0) countC++;
146         else countOther++;
147         temp = temp->next;
```




main.c

Output



```
148     }
149     printf("Block A: %d students\n",
           countA);
150     printf("Block B: %d students\n",
           countB);
151     printf("Block C: %d students\n",
           countC);
152     printf("Other Blocks: %d students\n",
           countOther);
153 }
154
155 int main() {
156     int choice, room;
157     char id[20], name[50], block[10];
158     Student* clonedList = NULL;
159
160     while (1) {
161         printf("\n--- Student Hostel
           Allotment System ---\n");
162         printf("1. Add New Allotment\n");
163         printf("2. Remove Student\n");
164         printf("3. Search by Name\n");
165         printf("4. Search by Room
           Number\n");
166         printf("5. Display Allotments
           Block-wise\n");
167         printf("6. Reverse Display\n");
168         printf("7. Clone List\n");
169         printf("8. Count Students per
           Block\n");
170         printf("9. Exit\n");
```




main.c

Output



```
171     printf("Enter your choice: ");
172     scanf("%d", &choice);
173
174     switch(choice) {
175         case 1:
176             printf("Enter Student ID:
177                     ");
178             scanf(" %[^\n]", id);
179
180             printf("Enter Name: ");
181             scanf(" %[^\n]", name);
182
183             printf("Enter Room Number:
184                     ");
185             scanf("%d", &room);
186
187             printf("Enter Block: ");
188             scanf(" %[^\n]", block);
189
190             addAllotment(id, name,
191                           room, block);
192             break;
193
194         case 2:
195             printf("Enter Student ID
196                     to remove: ");
197             scanf(" %[^\n]", id);
198             removeStudent(id);
199             break;
```



main.c

Output

`case 3:``printf("Enter Name to
search: ");``scanf("%[^\\n]", name);``searchByName(name);``break;``case 4:``printf("Enter Room Number
to search: ");``scanf("%d", &room);``searchByRoom(room);``break;`



main.c

Output



```
209 case 5:
210     printf("Enter Block to
211           display: ");
212     scanf("%[^\\n]", block);
213     displayBlockWise(block);
214     break;
215 case 6:
216     printf("Reverse Display
217           :\\n");
218     reverseDisplay(head);
219     break;
220 case 7:
221     clonedList = cloneList
222                 (head);
223     printf("List cloned
224           successfully.\\n");
225     break;
```



```
225         case 8:
226             countPerBlock();
227             break;
228
229         case 9:
230             exit(0);
231
232         default:
233             printf("Invalid choice.
234                 Try again.\n");
235     }
236     return 0;
237 }
```

Run



main.c

Output



```
--- Student Hostel Allotment System ---
```

1. Add New Allotment
2. Remove Student
3. Search by Name
4. Search by Room Number
5. Display Allotments Block-wise
6. Reverse Display
7. Clone List
8. Count Students per Block
9. Exit

```
Enter your choice: 1
```

```
Enter Student ID: 24BAD004
```

```
Enter Name: Arthi
```

```
Enter Room Number: 107
```

```
Enter Block: A
```

```
Allotment added successfully!
```

```
--- Student Hostel Allotment System ---
```

1. Add New Allotment
2. Remove Student
3. Search by Name
4. Search by Room Number
5. Display Allotments Block-wise
6. Reverse Display
7. Clone List
8. Count Students per Block
9. Exit

```
Enter your choice: 1
```

```
Enter Student ID: 24BAD019
```

```
Enter Name: Gabri
```

```
Enter Room Number: 207
```

```
Enter Block: A
```

```
Allotment added successfully!
```

```
--- Student Hostel Allotment System ---
```

1. Add New Allotment
2. Remove Student
3. Search by Name
4. Search by Room Number
5. Display Allotments Block-wise
6. Reverse Display
7. Clone List
8. Count Students per Block
9. Exit

```
Enter your choice: 1
```

```
Enter Student ID: 24CSE046
```

```
Enter Name: Yazhini
```

```
Enter Room Number: 101
```

```
Enter Block: B
```

```
Allotment added successfully!
```


--- Student Hostel Allotment System ---

1. Add New Allotment
2. Remove Student
3. Search by Name
4. Search by Room Number
5. Display Allotments Block-wise
6. Reverse Display
7. Clone List
8. Count Students per Block
9. Exit

Enter your choice: 3

Enter Name to search: Gabri

Found: ID=24BAD019, Name=Gabri, Room=207, Block=A

--- Student Hostel Allotment System ---

1. Add New Allotment
2. Remove Student
3. Search by Name
4. Search by Room Number
5. Display Allotments Block-wise
6. Reverse Display
7. Clone List
8. Count Students per Block
9. Exit

Enter your choice: 4

Enter Room Number to search: 105

No student found in room 105.

--- Student Hostel Allotment System ---

1. Add New Allotment
2. Remove Student
3. Search by Name
4. Search by Room Number
5. Display Allotments Block-wise
6. Reverse Display
7. Clone List
8. Count Students per Block
9. Exit

Enter your choice: 5

Enter Block to display: A

Students in Block A:

ID=24BAD004, Name=Arthi, Room=107

ID=24BAD019, Name=Gabri, Room=207