

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,
                             // not int
7      char name[50];
8      int roomNo;
9      char block[10];
10     struct Student* next;
11 } Student;
12
13 Student* head = NULL;
14
15 // Function to create new node
16 Student* createNode(char id[], char name[]
17                     , int room, char block[]) {
18     Student* newNode = (Student*)malloc
19                         (sizeof(Student));
20     strcpy(newNode->studentID, id);
21     strcpy(newNode->name, name);
22     newNode->roomNo = room;
23     strcpy(newNode->block, block);
24     newNode->next = NULL;
25     return newNode;
26 }
27
28 // Add new allotment
29 void addAllotment(char id[], char name[],
30                  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",
73 ▾                     temp->studentID, temp
                           ->name, temp->roomNo,
74 ▾                     temp->block);
75 ▾                 found = 1;
76 ▾             }
77             temp = temp->next;
78         }
79     }
80     if (!found) printf("No student found
                       in room %d.\n", room);
81 }
82
83 // Display block-wise
84 void displayBlockWise(char block[]) {
85     Student* temp = head;
86     int found = 0;
87     printf("Students in Block %s:\n",
```

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 }
118 // Count students per block
119 ▾ void countPerBlock() {
120     Student* temp = head;
121     int countA = 0, countB = 0, countC = 0
        , countOther = 0;
122 ▾     while (temp != NULL) {
123         if (strcmp(temp->block, "A") == 0)
            countA++;
124         else if (strcmp(temp->block, "B")
            == 0) countB++;
125         else if (strcmp(temp->block, "C")
            == 0) countC++;
126         else countOther++;
127         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



```
197         case 3:
198             printf("Enter Name to
199                     search: ");
200             scanf(" %[^\n]", name);
201             searchByName(name);
202             break;
203
204         case 4:
205             printf("Enter Room Number
206                     to search: ");
207             scanf("%d", &room);
208             searchByRoom(room);
209             break;
210
211         case 5:
212             printf("Enter Block to
213                     display: ");
214             scanf(" %[^\n]", block);
215             displayBlockWise(block);
216             break;
217
218         case 6:
219             printf("Reverse Display
220                     :\n");
221             reverseDisplay(head);
222             break;
223
224         case 7:
225             clonedList = cloneList
226                           (head);
227             printf("List cloned
228                     successfully.\n");
229             break;
```



main.c

Output



```
209         case 5:
210             printf("Enter Block to
                display: ");
211             scanf(" %[^\n]", block);
212             displayBlockWise(block);
213             break;
214
215         case 6:
216             printf("Reverse Display
                :\n");
217             reverseDisplay(head);
218             break;
219
220         case 7:
221             clonedList = cloneList
                (head);
222             printf("List cloned
                successfully.\n");
223             break;
224
225         case 8:
226             countPerBlock();
227             break;
228
229         case 9:
230             exit(0);
231
232         default:
233             printf("Invalid choice.
                Try again.\n");
234     }
235 }
236 return 0;
237 }
```

Run


```
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