Student hostel Allotment System.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Student {
    char studentID[20];
    char name[50];
    int roomNo;
    char block[10];
    struct Student* next;
} Student;

Student* head = NULL;

Student* createNode(char id[], char name[], int room,
                    char block[]) {
    Student* newNode = (Student*)malloc(sizeof(Student));
    strcpy(newNode->studentID, id);
    strcpy(newNode->name, name);
    newNode->roomNo = room;
    strcpy(newNode->block, block);
    newNode->next = NULL;
    return newNode;
}

void addAllotment(char id[], char name[], int room, char block[])
{
    Student* newNode = createNode(id, name, room, block);
```

```
if (head == NULL) {
    head = newNode;
} else {
    Student* temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
}
printf("Allotment added successfully!\n");
}

void removeStudent(char id[]) {
    if (head == NULL) {
        printf("No records found.\n");
        return;
    }
    Student *temp = head;
    Student *prev = NULL;
    while(temp != NULL && strcmp(temp->studentID, id) != 0)
{
    prev = temp;
    temp = temp->next;
}
if(temp == NULL) {
    printf("Student ID %s not found.\n", id);
    return;
}
if (prev == NULL) {
    head = head->next;
} else {
```

```
        prev -> next = temp -> next;
    }
    free(temp);
    printf("Student with ID %d removed successfully.\n", id);
}

void searchByName(char name[]) {
    Student* temp = head;
    int found = 0;
    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            printf("Found: ID = %d, Name = %s, Room = %d,
                Block = %s\n",
                temp->studentID, temp->name, temp->roomNo,
                temp->block);
            found = 1;
        }
        temp = temp->next;
    }
    if (!found) printf("No student with name %s found.
                \n", name);
}

void SearchByRoom(int room) {
    Student* temp = head;
    int found = 0;
    while (temp != NULL) {
        if (temp->roomNo == room) {
            printf("Found: ID = %d, Name = %s, Room = %d,
                Block = %s\n",
```

```
        temp->studentID, temp->name, temp->roomNo, temp->block);
        found = 1;
    }
    temp = temp->next;
}
if (!found) printf("No student found in room %d.\n", room);
}

void displayBlockWise(char block[]) {
    Student* temp = head;
    int found = 0;
    printf("Students in Block %s:\n", block);
    while (temp != NULL) {
        if (strcmp(temp->block, block) == 0) {
            printf("ID: %s, Name: %s, Room: %d\n",
                temp->studentID, temp->name, temp->roomNo);
            found = 1;
        }
        temp = temp->next;
    }
    if (!found) printf("No students in block %s.\n", block);
}

void reverseDisplay(Student* node) {
    if (node == NULL) return;
    reverseDisplay(node->next);
    printf("ID: %s, Name: %s, Room: %d, Block: %s\n",
        node->studentID, node->name, node->roomNo, node->block);
}
```

```
Student * cloneList (Student* head){
    if (head == NULL) return NULL;
    Student* newHead = NULL, * tail = NULL;
    Student * temp = head;
    while (temp != NULL) {
        Student * newNode = createNode(temp → studentID,
            temp → name, temp → roomNo, temp → block);
        if (newHead == NULL) {
            newHead = newNode;
            tail = newHead;
        } else {
            tail → next = newNode;
            tail = newNode;
        }
        temp = temp → next;
    }
    return newHead;
}

void countPerBlock() {
    Student* temp = head;
    int countA = 0, countB = 0, countC = 0, countOther = 0;
    while (temp != NULL) {
        if (strcmp (temp → block, "A") == 0) countA++;
        else if (strcmp (temp → block = "B") == 0) countB++;
        else if (strcmp (temp → block = "C") == 0) countC++;
        else countOther++;
```

```
    temp = temp → next ;
}
    printf (" Block A : %d students \n ", countA);
    printf (" Block B : %d students \n ", countB );
    printf (" Block C : %d students \n ", countC );
    printf (" Other Blocks : %d Students \n ", countOther );
}
int main() {
    int choice , room ;
    char id[20], name[50], block[10],
    Student * clonelist : NULL ;
    while (1) {
      printf (" \n --- Student Hostel Allotment System --- \n");
      printf (" 1. Add new Allotment \n " );
      printf (" 2. Remove Student \n");
      printf (" 3. Search by Name \n");
      printf (" 4. Search by Name \n");
      printf (" 5. Display Allotments Block - wise \n " );
      printf ("6. Reverse Display \n");
      printf (" 7. Clone List \n " );
      printf (" 8. Count Students per Block \n");
      printf (" 9. Exit \n" );
      printf (" Enter your choice;");
      Scanf (" %d ", & choice );
  Switch (choice) {
```

```
Case 1:
    printf("Enter Student ID: ");
    scanf("%d[^\n]", id);
    printf("Enter Name:");
    scanf("%[^\n]", name);
    printf("Enter Room Number:");
    scanf("%d", &room);
    printf("Enter Block: ");
    scanf("%[^\n]", block);

    addAllotment(id, name, room, block);
    break;

Case 2:
    printf("Enter the Student ID to remove:");
    scanf("%[^\n]", id);
    removeStudent(id);
    break;

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;
```

```
case 5 :
        printf ("Enter Block to display : ");
        scanf ("%d", &block);
        displayBlockWise (block);
        break;

case 6 :
        printf ("Reverse Display : \n");
        reverseDisplay (head);
        break;

case 7 :
        clonedList = cloneList (head);
        printf ("List cloned successfully. \n");
        break;

case 8 :
        CountPerBlock ();
        break;

case 9 :
        exit (0);

default :
        printf ("Invalid Choice . Try again . \n");

    }
}
    return 0;

}
```

Student hostel allotment system-71052410040 -Assignment1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Student {
   char studentID[20];   // now string, not int
   char name[50];
   int roomNo;
   char block[10];
   struct Student* next;
} Student;

Student* head = NULL;

// Function to create new node
Student* createNode(char id[], char name[], int room,
char block[]) {
   Student* newNode =
(Student*)malloc(sizeof(Student));
   strcpy(newNode->studentID, id);
   strcpy(newNode->name, name);
   newNode->roomNo = room;
   strcpy(newNode->block, block);
   newNode->next = NULL;
   return newNode;
}

// Add new allotment
void addAllotment(char id[], char name[], int room, char
block[]) {
   Student* newNode = createNode(id, name, room,
block);
   if (head == NULL) {
      head = newNode;
   } else {
      Student* temp = head;
      while (temp->next != NULL)
         temp = temp->next;
      temp->next = newNode;
   }
   printf("Allotment added successfully!\n");
}
```

```
// Remove student by ID
void removeStudent(char id[]) {
    if (head == NULL) {
        printf("No records found.\n");
        return;
    }
    Student* temp = head;
    Student* prev = NULL;
    while (temp != NULL && strcmp(temp->studentID, id) !
= 0) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Student ID %s not found.\n", id);
        return;
    }
    if (prev == NULL) {
        head = head->next;
    } else {
        prev->next = temp->next;
    }
    free(temp);
    printf("Student with ID %s removed successfully.\n",
id);
}

// Search by name
void searchByName(char name[]) {
    Student* temp = head;
    int found = 0;
    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            printf("Found: ID=%s, Name=%s, Room=%d,
Block=%s\n",
                   temp->studentID, temp->name,
temp->roomNo, temp->block);
            found = 1;
        }
        temp = temp->next;
    }
    if (!found) printf("No student with name %s found.\n",
name);
}

// Search by room number
void searchByRoom(int room) {
    Student* temp = head;
    int found = 0;
    while (temp != NULL) {
        if (temp->roomNo == room) {
            printf("Found: ID=%s, Name=%s, Room=%d,
Block=%s\n",
                   temp->studentID, temp->name,
temp->roomNo, temp->block);
            found = 1;
        }
        temp = temp->next;
    }
    if (!found) printf("No student found in room %d.\n",
room);
}
```

```
// Display block-wise
void displayBlockWise(char block[]) {
   Student* temp = head;
   int found = 0;
   printf("Students in Block %s:\n", block);
   while (temp != NULL) {
      if (strcmp(temp->block, block) == 0) {
         printf("ID=%s, Name=%s, Room=%d\n",
               temp->studentID, temp->name,
temp->roomNo);
         found = 1;
      }
      temp = temp->next;
   }
   if (!found) printf("No students in block %s.\n", block);
}

// Reverse display using recursion
void reverseDisplay(Student* node) {
   if (node == NULL) return;
   reverseDisplay(node->next);
   printf("ID=%s, Name=%s, Room=%d, Block=%s\n",
        node->studentID, node->name, node->roomNo,
node->block);
}

// Clone list
Student* cloneList(Student* head) {
   if (head == NULL) return NULL;
   Student* newHead = NULL, *tail = NULL;
   Student* temp = head;
   while (temp != NULL) {
      Student* newNode = createNode(temp->studentID,
temp->name, temp->roomNo, temp->block);
      if (newHead == NULL) {
         newHead = newNode;
         tail = newHead;
      } else {
         tail->next = newNode;
         tail = newNode;
      }
      temp = temp->next;
   }
   return newHead;
}

// Count students per block
void countPerBlock() {
   Student* temp = head;
   int countA = 0, countB = 0, countC = 0, countOther =
0;
   while (temp != NULL) {
      if (strcmp(temp->block, "A") == 0) countA++;
      else if (strcmp(temp->block, "B") == 0) countB++;
      else if (strcmp(temp->block, "C") == 0) countC++;
      else countOther++;
      temp = temp->next;
   }
   printf("Block A: %d students\n", countA);
   printf("Block B: %d students\n", countB);
   printf("Block C: %d students\n", countC);
   printf("Other Blocks: %d students\n", countOther);
}
```

```
int main() {
    int choice, room;
    char id[20], name[50], block[10];
    Student* clonedList = NULL;

    while (1) {
        printf("\n--- Student Hostel Allotment System
---\n");
        printf("1. Add New Allotment\n");
        printf("2. Remove Student\n");
        printf("3. Search by Name\n");
        printf("4. Search by Room Number\n");
        printf("5. Display Allotments Block-wise\n");
        printf("6. Reverse Display\n");
        printf("7. Clone List\n");
        printf("8. Count Students per Block\n");
        printf("9. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                printf("Enter Student ID: ");
                scanf(" %[^\n]", id);

                printf("Enter Name: ");
                scanf(" %[^\n]", name);

                printf("Enter Room Number: ");
                scanf("%d", &room);

                printf("Enter Block: ");
                scanf(" %[^\n]", block);

                addAllotment(id, name, room, block);
                break;

            case 2:
                printf("Enter Student ID to remove: ");
                scanf(" %[^\n]", id);
                removeStudent(id);
                break;

            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;

            case 5:
                printf("Enter Block to display: ");
                scanf(" %[^\n]", block);
                displayBlockWise(block);
                break;

            case 6:
                printf("Reverse Display:\n");
                reverseDisplay(head);
                break;

            case 7:
                clonedList = cloneList(head);
                printf("List cloned successfully.\n");
                break;

            case 8:
                countPerBlock();
                break;

            case 9:
                exit(0);

            default:
                printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

```
--- 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
```