Alumni Tracking System

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

typedef struct Alumni {
    int id;
    char name[50];
    char department[50];
    int graduationYear;
    char occupation[50];
    struct Alumni *next;
} Alumni;

// Function Prototypes
Alumni *CreateAlumni(int id, char name[], char dept[],
                     int year, char occ[]);
void insertAlumni(Alumni **head, int id, char name[],
                  char dept[], int year, char occ[]);
Alumni *SearchByID(Alumni *head, int id);
Alumni *searchByName(Alumni *head, const char *name);
void deleteByID(Alumni **head, int id);
void displayList(Alumni *head);
void reverseDisplay(Alumni *head);
void cloneList(Alumni *head, Alumni **cloneHead);
void departmentWiseList(Alumni *head, const char *dept);
```

```
void reverseList(Alumni **head);
void FreeList(Alumni **head);

int main(){
    Alumni *head = NULL, *cloneHead = NULL;
    int choice, id, year;
    char name[50], dept[50], occ[50];

    while(1){
        Printf("\n---- Alumni Tracking system ----\n");
        Printf("1. Insert Alumni\n2. search by ID\n3.
                Search by Name\n");
        Printf("4. Delete by ID\n5. Display List\n6. ReversedDisplay\n")
        Printf("7. Clone List\n8. Department-wise List\n9.
                Reverse Linked List\n10. Exit\n");
        Printf("Enter choice:");
        scanf("%d", &choice);

    switch(choice){
        case 1:
            Printf("Enter ID:");
            scanf("%d", &id);
            getchar();
```

```
Printf (" Enter Name :");
fgets (name, sizeof (name), stdin);
name [strcspn (name, "\n")] = 0;
Printf (" Enter Department: ");
fgets (dept, sizeof (dept), stdin);
dept [strcspn (dept, "\n")] = 0;
Printf (" Enter Graduation year ");
scanf (" %d", &year);
getchar();
Printf (" Enter occupation: ");
fgets (occ, sizeof (occ), stdin);
occ [strcspn (occ, "\n")] = 0;
insertAlumni (&head, id, name, dept, year, occ);
break;
```

Case 2:

```
Printf (" Enter ID to search :");
scanf (" %d", &id);
Alumni * foundID = searchByID (head, id);
if (foundID) {
    Printf (" Found : %d | %s | %s | %d | %s \n", foundID ->id,
        foundID -> name, foundID -> department, foundID
    -> graduationYear, foundID -> occupation);
}
```

```
else {
    Printf (" Alumni not found! \n");
break;
}

case 3:
    printf( "Enter name to search:");
    getchar();
    fgets ( name, sizeof (name), stdin);
    name [strcspn ( name, "\n")] = 0;
    Alumni * foundName = searchByName (head, name);
    if (foundName ){
        printf( "Founds : %d |%.s| %.s| %d |%.s \n",
    foundname -> id , foundName -> name, foundName -> department,
    foundName -> graduationYear , foundName -> occupation);
    else {
        Printf ( " Alumni not found! \n");
        break;
    }

case 4:
    Printf ( " Enter ID to delete : ");
    scanf ( "%d", &id);
    deleteByID ( &head, id);
    break;
```

```
case 5:
    displayList(head);
    break;

case 6:
    reverseDisplay(head);
    break;

case 7:
    cloneList(head, &cloneHead);
    printf("Cloned Lists:\n");
    displayList(cloneHead);
    break;

case 8:
    printf("Enter Department:");
    getchar();
    fgets(dept, sizeof(dept), stdin);
    dept[strcspn(dept, "\n")] = 0;
    departmentWiseList(head, dept);
    break;

case 9:
    reversedList(&head);
    printf("Linked list reversed!\n");
    break;
```

```
        (ase 10:
            freeList (&head);
            freeList (&cloneHead);
            printf (" Exiting... \n");
            exit (0);
        default:
            printf (" Invalid choice! \n");
    }
}
return 0;
}

// create new Alumni node
Alumni* createAlumni (int id, char name[], char dept[],
                      int year[], char occ[]){
    Alumni * newAlumni = (Alumni*) malloc (sizeof (Alumni));
    newAlumni -> id = id;
    strcpy (newAlumni -> name , name);
    strcpy (newAlumni -> department , dept);
    newAlumni -> graduationYear = year;
    strcpy (newAlumni -> occupation , occ);
    newAlumni -> next = NULL;
    return newAlumni;
}
```

```
// Insert at End with duplicate check

Void insertAlumni (Alumni ** head, int id, char name[],
                   char dept[], int year, char occ[]){

    if (searchByID (*head, id)){
        printf ("ID %d already exists! Insertion failed.\n", id);
        return;
    }

    Alumni *newAlumni = createAlumni (id, name, dept, year, occ);

    if (*head == NULL){
        *head = newAlumni;
        return;
    }

    Alumni *temp = *head;
    while (temp->next){
        temp = temp->next;
    }
    temp->next = newAlumni;
}

// Search by ID

Alumni * searchByID (Alumni *head, int id){
    while (head){
        if (head->id == id)
            return head;
```

```
        head = head -> next;
    }
    return NULL;
}

// Search by Name
Alumni * searchByName (Alumni *head, const char *name)
{
    while (head) {
        if (strcasecmp (head -> name, name) == 0) {
            return head;

// Delete Node by ID
void deleteBYID (Alumni **head, int id) {
    Alumni *temp = *head, * prev = NULL;
    while (temp && temp -> id != id) {
        prev = temp;
        temp = temp -> next;
    }
    if (!temp) {
        printf ("Record not found!\n");
        return;
    }
```

```
    if (prev){
        prev->next = temp -> next;
    }
    else{
        *head = temp -> next;
    free(temp);
    printf("Alumni with ID %d deleted successfully.\n",id);
}

// Display Full List
void displayList(Alumni *head){
    if (!head){
        printf("No Records.\n");
        return;
    }
    while (head){
        printf("%d | %s | %s | %d | %s\n", head->id,
        head->name, head->department, head->graduation,
        head->occupation);
        head = head->next;
    }
}
```

```
// Reversed Display using Recursion
void reverseDisplay (Alumni * head) {
    if (! head ) return ;
    reverseDisplay (head -> next);
    printf ("%d | %s | %s | %d | %s \n", head -> id,
    head -> name, head -> department, head -> graduationYear,
    head -> occupation );
}

// clone linked list
void cloneList (Alumni * head, Alumni ** cloneHead){
    * cloneHead = NULL;
    Alumni *tail = NULL;
    while (head){
        Alumni * newNode = createAlumni (head -> id,
    head -> name, head -> department, head -> graduationYear,
    head -> occupation );
        if (* cloneHead == NULL){
            * cloneHead = tail = newNode;
    }
    else {
        tail -> next = newNode;
```

```
        tail = accoNode;
    }
    head = head -> next;
  }
}

// Department-wise display
void departmentWiseList(Alumni *head, const char *dept){
    int found = 0;
    while (head){
        if (strcasecmp(head->department, dept) == 0){
            Printf("%d | %s | %s | %d | %s\n", head->id,
        head->name, head->department, head->graduationYear,
        head->occupation);
            found = 1;
    }
        head = head->next;
  }
  if (!found){
    Printf("No records found for Department: %s\n", dept);
}
```

```
// Reverse Linked list (actual reversal)
void reverseList(Alumni **head) {
    Alumni *prev = NULL, *curr = *head,
    *next = NULL;
    while (curr) {
        next = curr -> next;
        curr -> next = prev;
        prev = curr;
        curr = next;
    }
    *head = prev;
}

// Free all nodes
void freeList(Alumni **head) {
    Alumni *temp;
    while (*head) {
        temp = *head;
        *head = (*head) -> next;
        free(temp);
    }
}
```

main.c

Run

```c
// Create new Alumni node
Alumni* createAlumni(int id, char name[], char dept[], int year, char occ[]) {
    Alumni *newAlumni = (Alumni*)malloc(sizeof(Alumni));
    newAlumni->id = id;
    strcpy(newAlumni->name, name);
    strcpy(newAlumni->department, dept);
    newAlumni->graduationYear = year;
    strcpy(newAlumni->occupation, occ);
    newAlumni->next = NULL;
    return newAlumni;
}

// Insert at End
void insertAlumni(Alumni **head, int id, char name[], char dept[], int year, char occ[]) {
    Alumni *newAlumni = createAlumni(id, name, dept, year, occ);
    if (*head == NULL) {
        *head = newAlumni;
        return;
    }
    Alumni *temp = *head;
    while (temp->next)
        temp = temp->next;
    temp->next = newAlumni;
}

// Search by ID
Alumni* searchByID(Alumni *head, int id) {
    while (head) {
        if (head->id == id)
            return head;
        head = head->next;
    }
    return NULL;
}

// Search by Name
Alumni* searchByName(Alumni *head, const char *name) {
    while (head) {
        if (strcasecmp(head->name, name) == 0)
            return head;
        head = head->next;
    }
    return NULL;
}

// Delete Node by ID
void deleteByID(Alumni **head, int id) {
    Alumni *temp = *head, *prev = NULL;
    while (temp && temp->id != id) {
        prev = temp;
        temp = temp->next;
    }
    if (!temp) {
        printf("Record not found!\n");
        return;
    }
    if (prev)
        prev->next = temp->next;
    else
        *head = temp->next;
    free(temp);
    printf("Alumni with ID %d deleted successfully!\n", id);
}

// Display Full List
void displayList(Alumni *head) {
    if (!head) {
        printf("No Records!\n");
        return;
    }
    while (head) {
        printf("%d | %s | %s | %d | %s\n",
            head->id, head->name, head
```

Output

Clear

```
---- Alumni Tracking System ----
1. Insert Alumni
2. Search by ID
3. Search by Name
4. Delete by ID
5. Display List
6. Reverse Display
7. Clone List
8. Department-wise List
9. Exit
Enter choice: 1
Enter ID: 0010
Enter Name: Haripriya
Enter Department: AIML
Enter Graduation Year: 2028
Enter Occupation: Data Analytics

---- Alumni Tracking System ----
1. Insert Alumni
2. Search by ID
3. Search by Name
4. Delete by ID
5. Display List
6. Reverse Display
7. Clone List
8. Department-wise List
9. Exit
Enter choice: 1
Enter ID: 0011
Enter Name: Riya
Enter Department: AIML
Enter Graduation Year: 2028
Enter Occupation: Software developer

---- Alumni Tracking System ----
1. Insert Alumni
2. Search by ID
3. Search by Name
4. Delete by ID
5. Display List
6. Reverse Display
7. Clone List
8. Department-wise List
9. Exit
Enter choice: 2
Enter ID to search: 0010
Found: 10 | Haripriya  | AIML | 2028 | Data An


---- Alumni Tracking System ----
1. Insert Alumni
2. Search by ID
3. Search by Name
4. Delete by ID
5. Display List
6. Reverse Display
7. Clone List
8. Department-wise List
9. Exit
Enter choice:
```