THE NATIONAL INSTITUTE OF ENGINEERING

MYSURU-570008,

(An Autonomous Institute under VTU, Belagavi)

A
Report on

“PROJECT TITLE”
(MINI PROJECT- VOTING MANAGEMENT SYSTEM)

submitted in the partial fulfilment of the requirements for third semester in Bachelor of Engineering in Computer science & Engineering.

Submitted by

STUDENT NAME.	**USN**
MOHAMMED ARMAN ALI	4NI22CS123
MADAN A	4NI22CS111
HARSHITH M	4NI22CS071
HARSHITH M	4NI22CS072
HARSHA T D	4NI22CS070

Under the Guidance of

Mrs. AMRUTHA SREE
Asst. Professor Dept.
of CS&E.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MYSURU-570008, KARNATAKA, INDIA

2023-2024

CHAPTER 1: INTRODUCTION

A Study of Voting Management Systems Implemented in C Programming with Linked Lists.

A Voter Management System is a software application designed to manage and organize information related to voters in an election process. Using a linked list data structure for this system can be an effective way to store and manipulate voter data.

This project embarks on an exploration of voting management systems developed in C programming language, with a focus on their utilization of linked lists.

Linked lists offer dynamic memory allocation, flexibility, and efficient insertion and deletion of elements, making them suitable for scenarios where the number of voters may change during the election process.

Fundamental data structure consisting of nodes linked sequentially.

Enables dynamic memory allocation and efficient insertion/deletion of elements.

Enables quick and efficient addition or removal of voters during an election process.

Optimizes memory usage by allocating space only for actual data and references. Simplifies coding for common operations like insertion, deletion, and traversal.

At its essence, a voting management system implemented in C with linked lists operates as a sophisticated data structure, adept at organizing and manipulating voter data, candidate profiles, and ballot information.

CHAPTER 2: IMPLEMENTATION

The implementation of the system involves several key components:

The **voterinsertion()** function adds the new voter to the linked list.

The **searchvoter()** function searches the voter based on name and age.

The **voterdeletion()** function delets the voter based on name and age.

The **displayvoter()** function displays the name and age of the voter.

The **candidateinsertion()** function adds the new candidate to the linked list.

The **searchcandiidate()** function searches the voter based on name and age.

The **candidatedeletion**() function deletes the candidate based on name and age.

The **displaycandidate**() function displays the name and age of the candidate.

1. **Error Handling:**
    - Implement error handling mechanisms to handle invalid user inputs, such as invalid age of the voter to vote and invalid age of the candidate to contest in the election.
2. **Searching:**
    - Searching of voter and candidate based on age and id.

CODE:

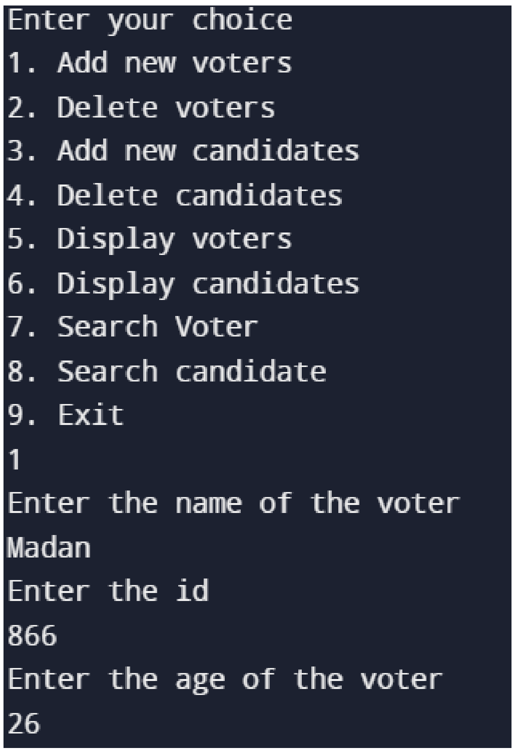
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Voter {
    char name[50];
    int id;
    int age;
    struct Voter* next;
} Voter;

typedef struct Candidate {
    char name[50];
    struct Candidate* next;
} Candidate;

Voter* voterinsertion(Voter*, Voter*, int, int);
Voter* voterdeletion(Voter*, char*, int, int);
Candidate* candidateinsertion(Candidate*, Candidate*);
Candidate* candidatedeletion(Candidate*, char*);
Voter* searchvoter(Voter*, char*, int);
Candidate* searchcandidate(Candidate*, char*);
void displayvoter(Voter*);
void displaycandidate(Candidate*);

int main() {
    Voter* start = NULL;
    Candidate* start1 = NULL;
    char v_name[100], c_name[100];
    int v_age, choice, key_age, c_age, keysearchv_id, id, key_id, keysearchv_age;
    char keyv_name[100], key_cname[100], keysearchv_name[100],
keysearchc_name[100];
    while (1) {
        printf("Enter your choice\n");
        printf("1. Add new voters\n");
        printf("2. Delete voters\n");
        printf("3. Add new candidates\n");
        printf("4. Delete candidates\n");
        printf("5. Display voters\n");
        printf("6. Display candidates\n");
        printf("7. Search Voter\n");
        printf("8. Search candidate\n");
        printf("9. Exit\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the name of the voter\n");
                scanf("%s", v_name);
                printf("Enter the id\n");
                scanf("%d", &id);
                printf("Enter the age of the voter\n");
                scanf("%d", &v_age);
                if (v_age >= 18) {
                    Voter* new_voter = (Voter*)malloc(sizeof(Voter));
                    strcpy(new_voter->name, v_name);
                    new_voter->id = id;
                    new_voter->age = v_age;
                    start = voterinsertion(start, new_voter, v_age, id);
                } else {
                    printf("Voter not eligible to vote\n");
                }
                break;
            case 2:
                printf("Enter voter name to delete\n");
                scanf("%s", keyv_name);
                printf("Enter the voter age\n");
                scanf("%d", &key_age);
                printf("Enter the ID:\n");
                scanf("%d", &key_id);
                start = voterdeletion(start, keyv_name, key_age, key_id);
                break;
            case 3:
                printf("Enter the name of the candidate\n");
                scanf("%s", c_name);
                printf("Enter the age of candidate\n");
```

```
                scanf("%d", &c_age);
                if (c_age >= 35) {
                    Candidate* new_candidate =
(Candidate*)malloc(sizeof(Candidate));
                    strcpy(new_candidate->name, c_name);
                    start1 = candidateinsertion(start1, new_candidate);
                } else {
                    printf("The candidate is not eligible\n");
                }
                break;
            case 4:
                printf("Enter the candidate to delete\n");
                scanf("%s", key_cname);
                start1 = candidatedeletion(start1, key_cname);
                break;
            case 5:
                printf("The list of voters are:\n");
                displayvoter(start);
                break;
            case 6:
                printf("The list of candidates are:\n");
                displaycandidate(start1);
                break;
            case 7:
                printf("Enter the voter to be searched:\n");
                scanf("%s", keysearchv_name);
                printf("Enter the ID of the voter\n");
                scanf("%d", &keysearchv_id);
                searchvoter(start, keysearchv_name, keysearchv_id);
                break;
            case 8:
                printf("Enter the candidate to be searched:\n");
                scanf("%s", keysearchc_name);
                searchcandidate(start1, keysearchc_name);
                break;
            case 9:
                exit(0);
        }
    }
}
```

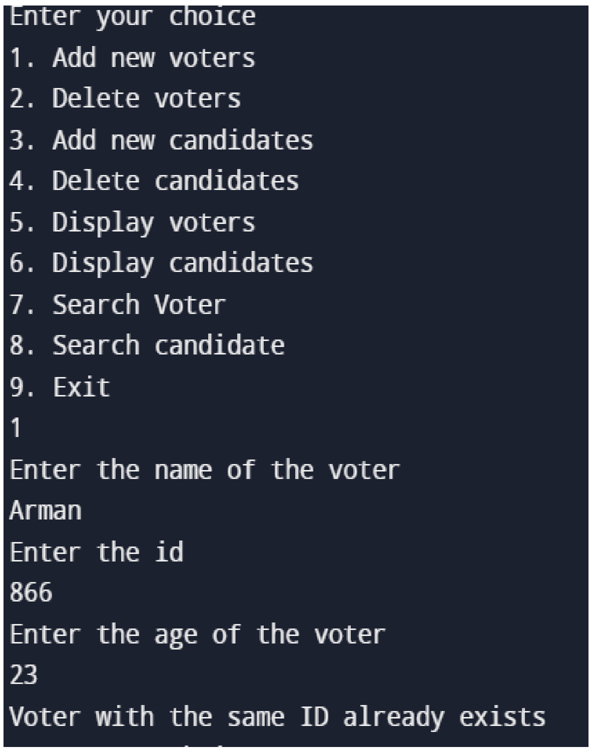
```
Voter* voterinsertion(Voter* start, Voter* new_voter, int x, int i) {
   // Check if the ID already exists
   Voter* ptr = start;
   while (ptr != NULL) {
      if (ptr->id == i) {
         printf("Voter with the same ID already exists\n");
         free(new_voter); // Free the memory allocated for the new voter
         return start;
      }
      ptr = ptr->next;
   }

   // Insert the new voter at the beginning
   new_voter->next = start;
   start = new_voter;
   return start;
}

Voter* voterdeletion(Voter* start, char* str1, int y, int d) {
   Voter *temp, *ptr, *prev;

   if (start == NULL) {
      printf("No voters are present\n");
      return start;
   }

   if ((strcmp(start->name, str1) == 0) && (start->age == y) && (start->id ==
d)) {
      printf("The voter deleted is: %s of ID: %d\n", str1, d);
      ptr = start;
      start = start->next;
      free(ptr);
      return start;
   }

   prev = NULL;
   ptr = start;
   while (ptr != NULL && (strcmp(ptr->name, str1) != 0 || ptr->id != d)) {
      prev = ptr;
```

```
        ptr = ptr->next;
    }

    if (ptr == NULL) {
        printf("Searched voter is not present\n");
        return start;
    }

    if (prev != NULL) {
        temp = prev->next;
        if (temp != NULL) {
            prev->next = temp->next;
            free(temp);
        }
    } else {
        start = ptr->next;
        free(ptr);
    }

    printf("Deleted voter is: %s of age: %d\n", ptr->name, ptr->age);
    return start;
}

Candidate* candidateinsertion(Candidate* start1, Candidate* new_candidate) {
    // Insert the new candidate at the beginning
    new_candidate->next = start1;
    return new_candidate;
}

Candidate* candidatedeletion(Candidate* start1, char* str3) {
    Candidate *temp, *ptr;

    if (start1 == NULL) {
        printf("No candidates are present\n");
        return start1;
    }

    if ((strcmp(start1->name, str3) == 0)) {
        printf("The candidate deleted is: %s\n", str3);
        ptr = start1;
        start1 = start1->next;
        free(ptr);
        return start1;
    } else {
        ptr = start1;
        while ((ptr != NULL) && (strcmp(ptr->name, str3) != 0)) {
            temp = ptr;
            ptr = ptr->next;
        }
        if (ptr == NULL) {
            printf("Searched candidate is not present\n");
        } else {
            temp->next = ptr->next;
            printf("Deleted candidate is: %s\n", ptr->name);
            free(ptr);
        }
    }
    return start1;
}

void displayvoter(Voter* start) {
    Voter* ptr;
    if (start == NULL)
        printf("No voters in the list\n");
    else {
        ptr = start;
        while (ptr != NULL) {
            printf("%s : age: %d\n", ptr->name, ptr->age);
            ptr = ptr->next;
        }
    }
}

void displaycandidate(Candidate* start1) {
    Candidate* ptr;
    if (start1 == NULL)
        printf("No candidates in the list\n");
    else {
        ptr = start1;
        while (ptr != NULL) {
            printf("%s\n", ptr->name);
            ptr = ptr->next;
        }
    }
}

Voter* searchvoter(Voter* start, char* str4, int e) {
    Voter* ptr = start;
    while (ptr != NULL) {
        if (strcmp(ptr->name, str4) == 0 && ptr->id == e) {
            printf("Voter found: %s, ID: %d\n", ptr->name, ptr->id);
            return ptr;
        }
        ptr = ptr->next;
    }
    printf("Voter not found\n");
    return NULL;
}

Candidate* searchcandidate(Candidate* start1, char* str5) {
    Candidate* ptr = start1;
    while (ptr != NULL) {
        if (strcmp(ptr->name, str5) == 0) {
            printf("Candidate found: %s\n", ptr->name);
            return ptr;
        }
        ptr = ptr->next;
    }
    printf("Candidate not found\n");
    return NULL;
}
```

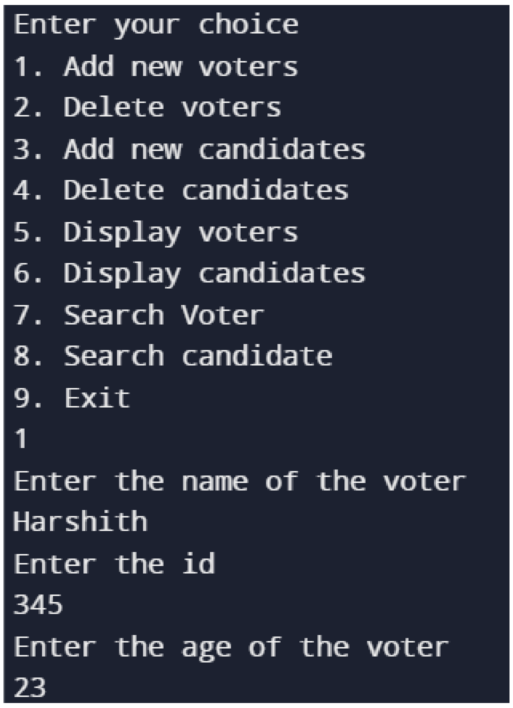
CHAPTER 3: RESULT



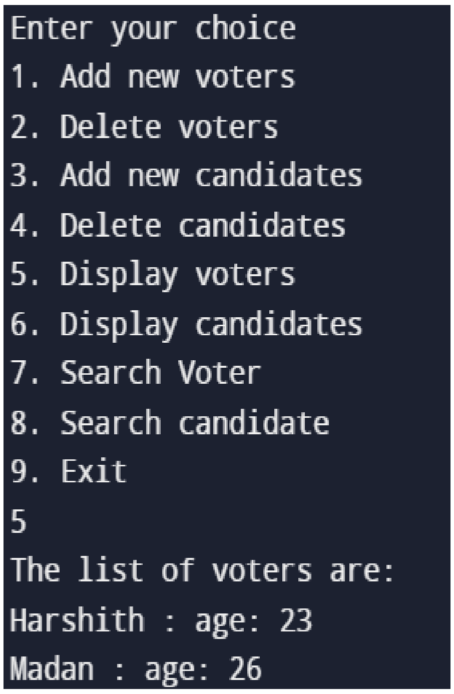
```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
1
Enter the name of the voter
Madan
Enter the id
866
Enter the age of the voter
26
```



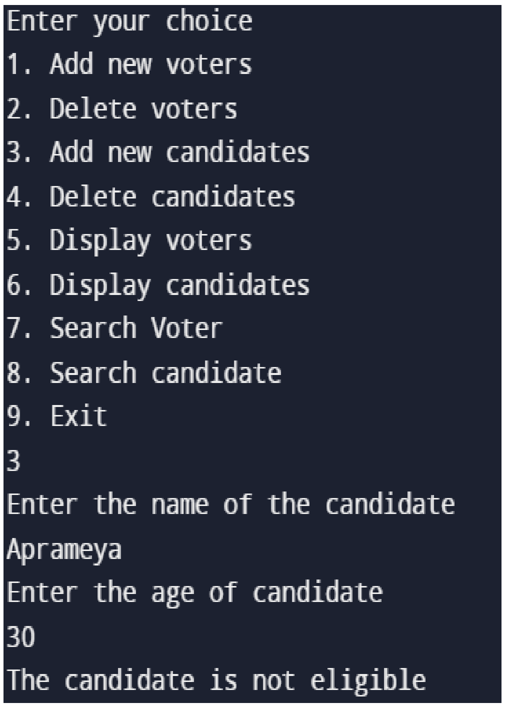
```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
1
Enter the name of the voter
Arman
Enter the id
866
Enter the age of the voter
23
Voter with the same ID already exists
```



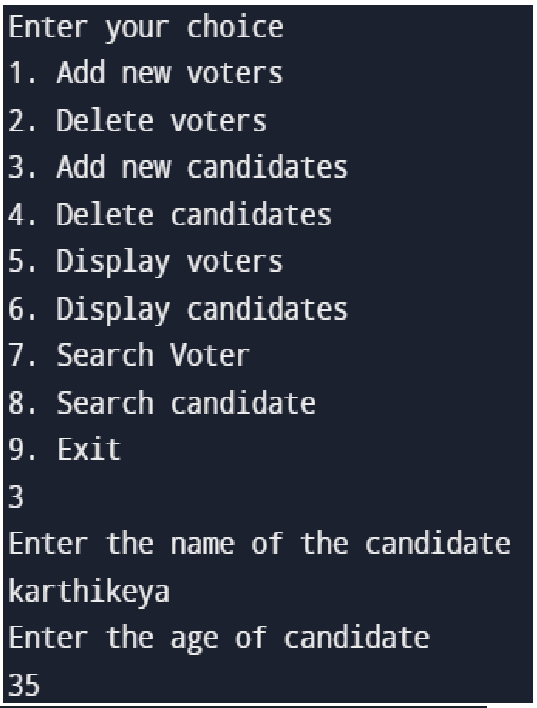
```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
1
Enter the name of the voter
Harshith
Enter the id
345
Enter the age of the voter
23
```



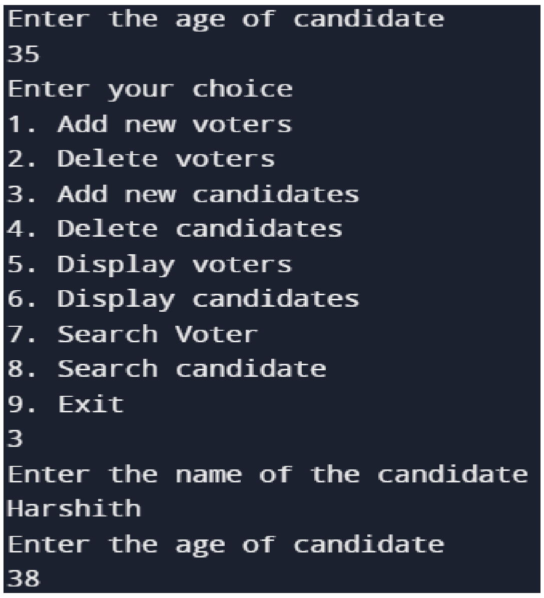
```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
5
The list of voters are:
Harshith : age: 23
Madan : age: 26
```



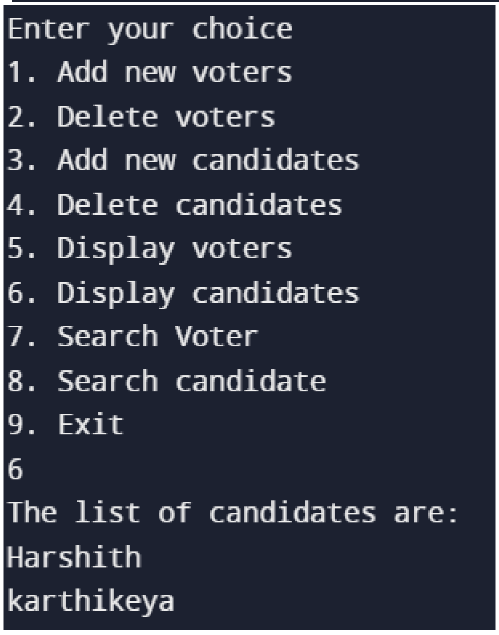
```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
3
Enter the name of the candidate
Aprameya
Enter the age of candidate
30
The candidate is not eligible
```



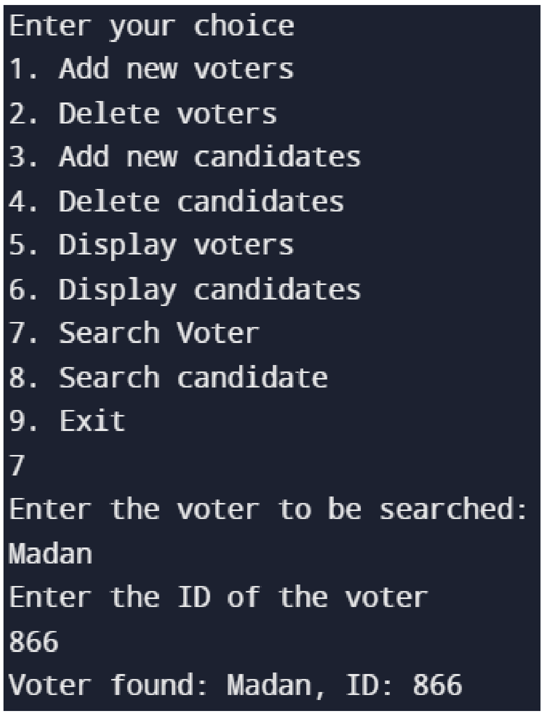
```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
3
Enter the name of the candidate
karthikeya
Enter the age of candidate
35
```



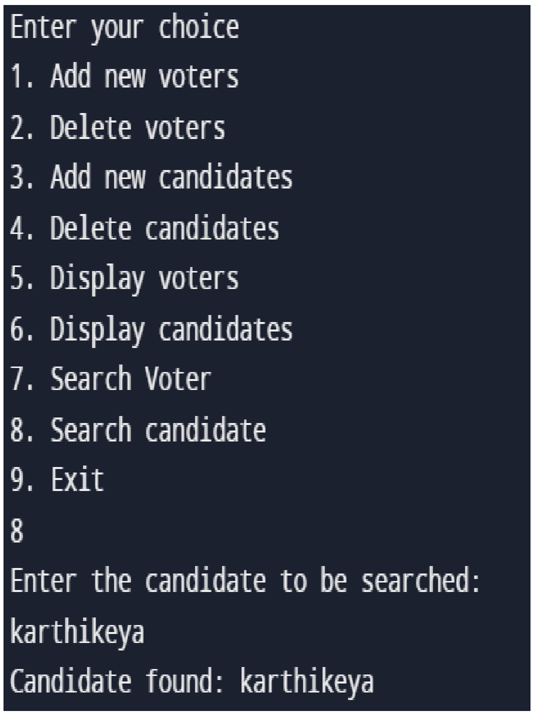
```
Enter the age of candidate
35
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
3
Enter the name of the candidate
Harshith
Enter the age of candidate
38
```



```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
6
The list of candidates are:
Harshith
karthikeya
```



```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
7
Enter the voter to be searched:
Madan
Enter the ID of the voter
866
Voter found: Madan, ID: 866
```



```
Enter your choice
1. Add new voters
2. Delete voters
3. Add new candidates
4. Delete candidates
5. Display voters
6. Display candidates
7. Search Voter
8. Search candidate
9. Exit
8
Enter the candidate to be searched:
karthikeya
Candidate found: karthikeya
```

CONCLUSION

A voting management system based on linked lists offers a robust and scalable solution for handling the complexities of the voting process, ensuring transparency, efficiency, and integrity in electoral procedures.

Overall, by leveraging the features of linked lists in a C program, a voting management system can be developed that is robust, efficient, and adaptable to the needs of electoral procedures, contributing to fair and transparent elections.

THANK YOU