**Name**: Ravi Jamanbhai Makwana

**Roll No**: 21BCP418

Div-6, G-12

**Assignment-4**

**Aim:**

Write a program to create City Database using Linked List.

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

typedef struct city

{

char name[20];

int x, y;

struct city \*next;

} city;

city \*head = NULL;

int counter = 0;

void add\_city()

{

city \*new\_city = malloc(sizeof(city));

printf("\nEnter name of city: ");

scanf("%s", &new\_city->name);

printf("Enter x coordinate: ");

scanf("%d", &new\_city->x);

printf("Enter y coordinate: ");

scanf("%d", &new\_city->y);

new\_city->next = NULL;

counter++;

if (head == NULL)

{

head = new\_city;

}

else

{

city \*temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = new\_city;

}

}

void print\_cities()

{

int i = 1;

city \*temp = head;

if (temp == NULL)

{

printf("\nNo cities are there in database!!!\n");

}

while (temp != NULL)

{

printf("\n---------------CITY %d---------------", i);

printf("\nName: %s", temp->name);

printf("\nX : %d", temp->x);

printf("\nY : %d\n", temp->y);

temp = temp->next;

i++;

}

}

void print\_given\_city(city \*temp)

{

printf("\n-----------------------------------------");

printf("\nName: %s", temp->name);

printf("\nX : %d", temp->x);

printf("\nY : %d\n", temp->y);

}

void search\_city\_by\_name()

{

char name[20];

printf("\nEnter name of city: ");

scanf("%s", &name);

city \*temp = head;

while (temp != NULL)

{

if (strcmp(temp->name, name) == 0)

{

print\_given\_city(temp);

return;

}

temp = temp->next;

}

printf("\nGiven city not found\n");

}

void search\_city\_by\_x\_y()

{

int x, y;

printf("\nEnter x coordinate: ");

scanf("%d", &x);

printf("Enter y coordinate: ");

scanf("%d", &y);

city \*temp = head;

while (temp != NULL)

{

if (temp->x == x && temp->y == y)

{

print\_given\_city(temp);

return;

}

temp = temp->next;

}

printf("\nGiven city not found\n");

}

float calculate\_distance(city \*temp1, int x, int y)

{

float distance = sqrt(pow(x - temp1->x, 2) + pow(y - temp1->y, 2));

return distance;

}

void delete\_city()

{

int flag = 0;

char name[20];

printf("\nEnter name of city: ");

scanf("%s", &name);

city \*temp = head;

city \*prev = NULL;

while (temp != NULL)

{

if (strcmp(temp->name, name) == 0)

{

if (prev == NULL)

{

head = temp->next;

}

else

{

prev->next = temp->next;

}

free(temp);

counter--;

flag++;

break;

}

prev = temp;

temp = temp->next;

}

if (flag == 0)

{

printf("\nGiven city not found\n");

}

}

void print\_city\_within\_distance()

{

int x, y;

float distance;

int flag = 0;

printf("\nEnter x coordinate: ");

scanf("%d", &x);

printf("Enter y coordinate: ");

scanf("%d", &y);

printf("Enter distance: ");

scanf("%f", &distance);

city \*temp = head;

while (temp != NULL)

{

if (calculate\_distance(temp, x, y) <= distance)

{

print\_given\_city(temp);

flag++;

}

temp = temp->next;

}

if (flag == 0)

{

printf("\nNo city found within given distance\n");

}

}

int main()

{

int choice;

while (1)

{

printf("\n1. Add city");

printf("\n2. Print all cities");

printf("\n3. Search city by name");

printf("\n4. Search city by x and y coordinate");

printf("\n5. Delete city");

printf("\n6. Print cities within given distance");

printf("\n7. Exit");

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

add\_city();

break;

case 2:

print\_cities();

break;

case 3:

search\_city\_by\_name();

break;

case 4:

search\_city\_by\_x\_y();

break;

case 5:

delete\_city();

break;

case 6:

print\_city\_within\_distance();

break;

case 7:

exit(0);

default:

printf("\nInvalid choice\n");

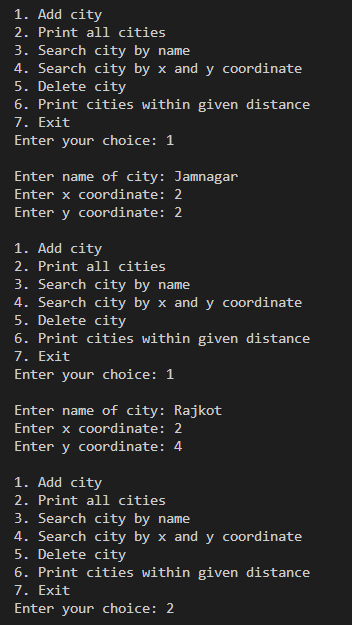
}

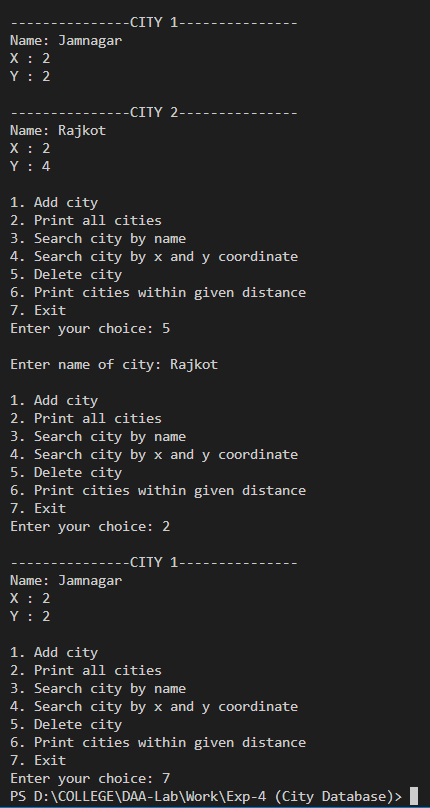
}

return 0;

}

**Output:**





**Analysis:**

**Time complexity Analysis:**

Implement the database using an array-based list implementation, and then a linked list implementation. Perform following analysis:

**a) Collect running time statistics for each operation in both**

**implementations.**

Collect running time statistics for each operation in both implementations:

• **Insert Node: O (1)**

We just need to add one data at starting by making the new node as head.

**• Search By Name: O (n)**

here let n = number of cities in the database we have to traverse the database until we find the city

**• Search By Coordinates: O (n)**

here let n = number of cities in the database we have to traverse the database until we find the city

**• Delete By Name: O (n)**

we have to first traverse to the given city and then we will delete the node

**• Delete By Coordinates: O (n)**

we have to first traverse to the given Coordinates and then we will delete the node

**• Distance Search City: O (n)**

we have to traverse to all the nodes of database to check that distance

**• Display: O (n)**

we have to traverse to all the nodes of database to print all data

**b) What are your conclusions about the relative advantages and**

**disadvantages of the two implementations?**

In the array implementation, if array gets overflow condition, then we have to create the array of double size and then we have to copy the previous array into it. So, this implementation its costly, but in linked list implementation it is not possible to overflow the linked list, so that will be very easy and less cost implementation.

**c) Would storing records on the list in alphabetical order by city name speed any of the operations?**

Storing string in alphabetical order can help us to reach city by name. It will be easy to reach to the city directly by its initial character.

**d) Would keeping the list in alphabetical order slow any of the operations?**

Yes, storing in the alphabetical order will slow down the operation of the insertion of the record. Because while inserting, first we have to search to the correct position and then we can insert the records. So, from 0(1) complexity will go to O(n) in that case.