EXPERIMENT 4

20CP209P – Design and Analysis of Algorithm Lab

Aim:

Implement a city database using unordered lists. Each database record contains the name of the city (a string of arbitrary length) and the coordinates of the city expressed as integer x and y coordinates. Your program should allow following functionalities:

a) Insert a record,

b) Delete a record by name or coordinate,

c) Search a record by name or coordinate.

d) Pint all records within a given distance of a specified point.

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.

b) What are your conclusions about the relative advantages and disadvantages of the two implementations?

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

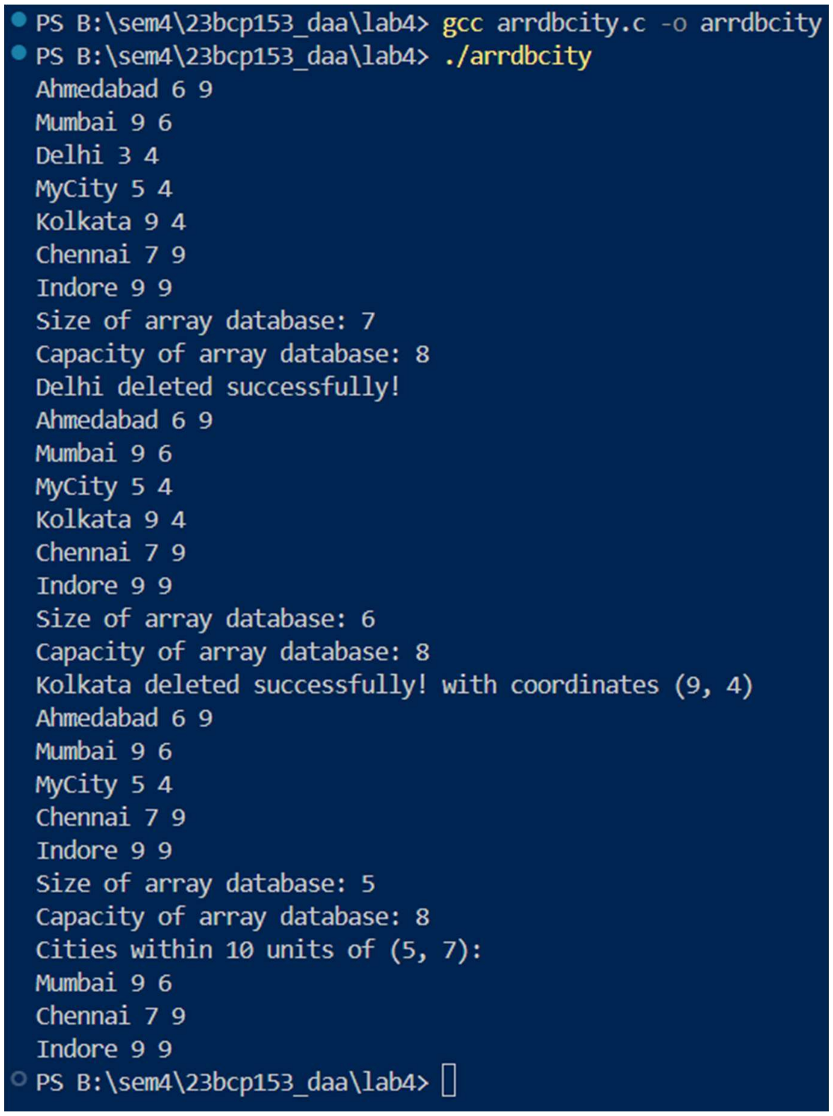
d) Would keeping the list in alphabetical order slow any of the operations?

Code:

Array-Based:

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

#include <math.h>

typedef struct City {
   char* name;
   int x;
   int y;
} City;

typedef struct Arrdb {
    City** citiesarr;
    int size;
    int capacity;
} Arrdb;

City* create_city(char* name, int x, int y);
Arrdb* init_arr_db(int int_cap);
void insert_city_arr(Arrdb* arrdb, City* city);
void delete_by_name(Arrdb* arrdb, char* name);
void delete_by_coordinates(Arrdb* arrdb, int x, int y);
void print_arr_db(Arrdb* arrdb);
void print_within_dist(Arrdb* arrdb, int x, int y, int dist);

int main(void)
{

    Arrdb* arrdb = init_arr_db(2);
    City* city1 = create_city("Ahmedabad", 6, 9);
    insert_city_arr(arrdb, city1);
    City* city2 = create_city("Mumbai", 9, 6);
    insert_city_arr(arrdb, city2);
    City* city3 = create_city("Delhi", 3, 4);
    insert_city_arr(arrdb, city3);
    City* city4 = create_city("MyCity", 5, 4);
    insert_city_arr(arrdb, city4);
    City* city5 = create_city("Kolkata", 9, 4);
    insert_city_arr(arrdb, city5);
    City* city6 = create_city("Chennai", 7, 9);
    insert_city_arr(arrdb, city6);
    City* city7 = create_city("Indore", 9, 9);
    insert_city_arr(arrdb, city7);

    print_arr_db(arrdb);
    delete_by_name(arrdb, "Delhi");
    print_arr_db(arrdb);
    delete_by_coordinates(arrdb, 9, 4);
    print_arr_db(arrdb);

    print_within_dist(arrdb, 5, 7, 10);

    return 0;
}

City* create_city(char* name, int x, int y)
{
    City* city = (City*)malloc(sizeof(City));
    city->name = (char*)malloc(strlen(name) + 1);
    city->name = name;
    city->x = x;
    city->y = y;
    return city;
}

Arrdb* init_arr_db(int init_cap)
{
    Arrdb* arrdb = (Arrdb*)malloc(sizeof(Arrdb));
    arrdb->citiesarr = (City**)malloc(sizeof(City*) * init_cap);
    arrdb->size = 0;
    arrdb->capacity = init_cap;
}

void insert_city_arr(Arrdb* arrdb, City* city)
{
    if (arrdb->size == arrdb->capacity)
    {
        arrdb->citiesarr = (City**)realloc(arrdb->citiesarr, sizeof(City*) * arrdb->capacity * 2);
        arrdb->capacity *= 2;
    }
    arrdb->citiesarr[arrdb->size] = city;
    arrdb->size++;
}

void delete_by_name(Arrdb* arrdb, char* name)
{
    if (arrdb->size == 0)
    {
        printf("Database is already empyt!");
        return;
    }

    for (int i = 0; i < arrdb->size - 1; i++)
    {
        if (strcmp(arrdb->citiesarr[i]->name, name) == 0)
        {
            free(arrdb->citiesarr[i]->name);
            // free(arrdb->citiesarr[i]->x);
            // free(arrdb->citiesarr[i]->y);
            // can't do the above as have not done malloc for the above i.e. not dynamically allocated (not
pointers)
            free(arrdb->citiesarr[i]);

            for (int j = i; j < arrdb->size - 1; j++)
            {
                arrdb->citiesarr[j] = arrdb->citiesarr[j + 1];
            }
            arrdb->size--;

            printf("%s deleted successfully!\n", name);
            return;
        }
    }

    printf("City not found!\n");
    return;
}

void delete_by_coordinates(Arrdb* arrdb, int x, int y)
{
    if (arrdb->size == 0)
    {
        printf("Database is already empyt!");
        return;
    }

    for (int i = 0; i < arrdb->size - 1; i++)
    {
        if (arrdb->citiesarr[i]->x == x && arrdb->citiesarr[i]->y == y)
        {
            char* name = arrdb->citiesarr[i]->name;
            free(arrdb->citiesarr[i]->name);
            free(arrdb->citiesarr[i]);

            for (int j = i; j < arrdb->size - 1; j++)
            {
                arrdb->citiesarr[j] = arrdb->citiesarr[j + 1];
            }
            arrdb->size--;

            printf("%s deleted successfully! with coordinates (%d, %d)\n", name, x, y);
            return;
        }
    }
    return;
}

void print_arr_db(Arrdb* arrdb)
{
    for (int i = 0; i < arrdb->size; i++)
    {
        printf("%s %d %d\n", arrdb->citiesarr[i]->name, arrdb->citiesarr[i]->x, arrdb->citiesarr[i]->y);
    }
    printf("Size of array database: %d\n", arrdb->size);
    printf("Capacity of array database: %d\n", arrdb->capacity);
}

void print_within_dist(Arrdb* arrdb, int x, int y, int dist)
{
    if (arrdb->size == 0)
    {
        printf("Empty database");
        return;
    }

    printf("Cities within %d units of (%d, %d):\n", dist, x, y);

    for (int i = 0; i < arrdb->size; i++)
    {
        int diffx = arrdb->citiesarr[i]->x -x;
        int diffy = arrdb->citiesarr[i]->y -y;

        float distance = sqrt(pow(diffx, 2) - pow(diffy, 2));

        if (distance <= dist)
        {
            printf("%s %d %d\n", arrdb->citiesarr[i]->name, arrdb->citiesarr[i]->x, arrdb->citiesarr[i]->y);
        }
    }
}
```

Output:



```
PS B:\sem4\23bcp153_daa\lab4> gcc arrdbcity.c -o arrdbcity
PS B:\sem4\23bcp153_daa\lab4> ./arrdbcity
Ahmedabad 6 9
Mumbai 9 6
Delhi 3 4
MyCity 5 4
Kolkata 9 4
Chennai 7 9
Indore 9 9
Size of array database: 7
Capacity of array database: 8
Delhi deleted successfully!
Ahmedabad 6 9
Mumbai 9 6
MyCity 5 4
Kolkata 9 4
Chennai 7 9
Indore 9 9
Size of array database: 6
Capacity of array database: 8
Kolkata deleted successfully! with coordinates (9, 4)
Ahmedabad 6 9
Mumbai 9 6
MyCity 5 4
Chennai 7 9
Indore 9 9
Size of array database: 5
Capacity of array database: 8
Cities within 10 units of (5, 7):
Mumbai 9 6
Chennai 7 9
Indore 9 9
PS B:\sem4\23bcp153_daa\lab4>
```

Code:

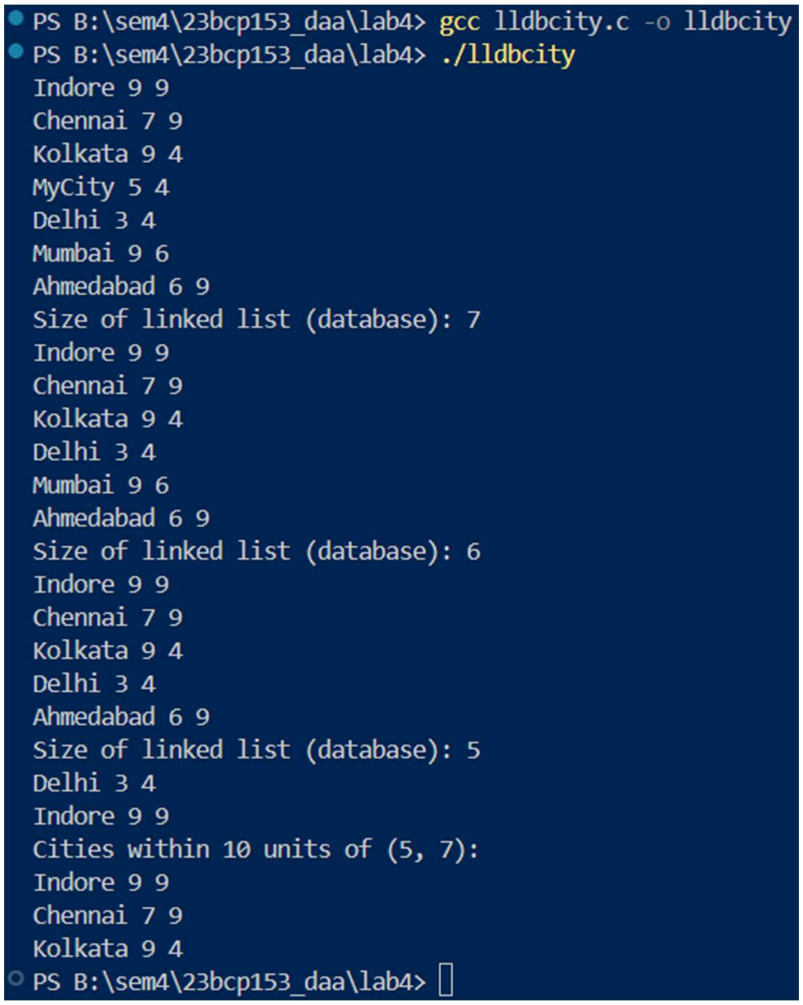
Linked-List-Based:

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

#include <math.h>

typedef struct City {
    char* name;
    int x;
    int y;
} City;
```

```
typedef struct Node {
   City* city;
   struct Node* next;
} Node;

typedef struct Lldb {
   Node* head;
   int size;
} Lldb;

City* create_city(char* name, int x, int y);
Node* create_node(City* city);
void insert_city_ll (Lldb* lldb, City* city);
void print_ll_db(Lldb* lldb);
void delete_by_name(Lldb* lldb, char* name);
void delete_by_coordinates(Lldb* lldb, int x, int y);
Node* search_by_name(Lldb* lldb, char* name);
Node* search_by_coordinates(Lldb* lldb, int x, int y);
void print_within_dist(Lldb* lldb, int x, int y, int dist);

int main(void)
{

   Lldb* lldb = (Lldb*)malloc(sizeof(Lldb));
   lldb->head = NULL;
   lldb->size = 0;

   City* city1 = create_city("Ahmedabad", 6, 9);
   insert_city_ll(lldb, city1);
   City* city2 = create_city("Mumbai", 9, 6);
   insert_city_ll(lldb, city2);
   City* city3 = create_city("Delhi", 3, 4);
   insert_city_ll(lldb, city3);
   City* city4 = create_city("MyCity", 5, 4);
   insert_city_ll(lldb, city4);
   City* city5 = create_city("Kolkata", 9, 4);
   insert_city_ll(lldb, city5);
   City* city6 = create_city("Chennai", 7, 9);
   insert_city_ll(lldb, city6);
   City* city7 = create_city("Indore", 9, 9);
   insert_city_ll(lldb, city7);

   print_ll_db(lldb);
   delete_by_name(lldb, "MyCity");
   print_ll_db(lldb);
   delete_by_coordinates(lldb, 9, 6);
   print_ll_db(lldb);
   Node* somecity = search_by_name(lldb, "Delhi");
   printf("%s %d %d\n", somecity->city->name, somecity->city->x, somecity->city->y);
   somecity = search_by_coordinates(lldb, 9, 9);
```

```
    printf("%s %d %d\n", somecity->city->name, somecity->city->x, somecity->city->y);

    print_within_dist(lldb, 5, 7, 10);

    return 0;
}

City* create_city(char* name, int x, int y)
{
    City* city = (City*)malloc(sizeof(City));
    city->name = (char*)malloc(strlen(name) + 1);
    city->name = name;
    city->x = x;
    city->y = y;
    return city;
}

Node* create_node(City* city)
{
    Node* node = (Node*) malloc(sizeof(Node));
    node->city = city;
    node->next = NULL;
    return node;
}

void insert_city_ll (Lldb* lldb, City* city)
{
    Node* node = create_node(city);
    node->next = lldb->head;
    lldb->head = node;
    lldb->size++;
}

void print_ll_db(Lldb* lldb)
{
    Node* trav = lldb->head;
    while(trav != NULL)
    {
        printf("%s %d %d\n", trav->city->name, trav->city->x, trav->city->y);
        trav = trav->next;
    }
    printf("Size of linked list (database): %d\n", lldb->size);
}

void delete_by_name(Lldb* lldb, char* name)
{
    Node* trav = lldb->head;
    if (strcmp(trav->city->name, name) == 0)
    {
        // if city is at head
        lldb->head = trav->next;
        free(trav);
        lldb->size--;
        return;
    }

    while (trav || strcmp(trav->next->city->name, name) != 0)
    {
        if (trav->next == NULL)
        {
            printf("City not found\n");
            return;
        }
        if (strcmp(trav->next->city->name, name) == 0)
        {
            Node* temp = trav->next;
            trav->next = trav->next->next;
            free(temp);
            lldb->size--;
            return;
        }

        trav = trav->next;
    }

    lldb->size--;

    return;
}

void delete_by_coordinates(Lldb* lldb, int x, int y)
{
    Node* trav = lldb->head;
    if (trav->city->x == x && trav->city->y == y)
    {
        // if city is at head
        lldb->head = trav->next;
        free(trav);
        lldb->size--;
        return;
    }

    while (trav || (trav->next->city->x != x && trav->next->city->y != y))
    {
        if (trav->next == NULL)
        {
            printf("City not found\n");
            return;
        }
        if (trav->next->city->x == x && trav->next->city->y == y)
        {
            Node* temp = trav->next;
            trav->next = trav->next->next;
            free(temp);
            lldb->size--;
            return;
        }

        trav = trav->next;
    }

    lldb->size--;

    return;
}

Node* search_by_name(Lldb* lldb, char* name)
{
    Node* trav = lldb->head;
    while (trav || strcmp(trav->city->name, name) != 0)
    {
        if (trav->next == NULL)
        {
            printf("City not found\n");
            return NULL;
        }
        if (strcmp(trav->city->name, name) == 0)
        {
            return trav;
        }

        trav = trav->next;
    }

    return NULL;
}

Node* search_by_coordinates(Lldb* lldb, int x, int y)
{
    Node* trav = lldb->head;
    while (trav || (trav->city->x != x && trav->city->y != y))
    {
        if (trav->next == NULL)
        {
            printf("City not found\n");
            return NULL;
        }
        if (trav->city->x == x && trav->city->y == y)
        {
            return trav;
        }

        trav = trav->next;
    }

    return NULL;
}

void print_within_dist(Lldb* lldb, int x, int y, int dist)
{
    if (lldb->head == NULL)
    {
        printf("Database Empyt\n");
        return;
    }

    printf("Cities within %d units of (%d, %d):\n", dist, x, y);

    Node* trav = lldb->head;
    while (trav)
    {
        int diffx = trav->city->x - x;
        int diffy = trav->city->y - y;

        float distance = sqrt(pow(diffx, 2) - pow(diffy, 2));

        if (distance <= dist)
        {
            printf("%s %d %d\n",  trav->city->name, trav->city->x, trav->city->y);
        }

        trav = trav->next;
    }
}
```

Output:



```
PS B:\sem4\23bcp153_daa\lab4> gcc lldbcity.c -o lldbcity
PS B:\sem4\23bcp153_daa\lab4> ./lldbcity
Indore 9 9
Chennai 7 9
Kolkata 9 4
MyCity 5 4
Delhi 3 4
Mumbai 9 6
Ahmedabad 6 9
Size of linked list (database): 7
Indore 9 9
Chennai 7 9
Kolkata 9 4
Delhi 3 4
Mumbai 9 6
Ahmedabad 6 9
Size of linked list (database): 6
Indore 9 9
Chennai 7 9
Kolkata 9 4
Delhi 3 4
Ahmedabad 6 9
Size of linked list (database): 5
Delhi 3 4
Indore 9 9
Cities within 10 units of (5, 7):
Indore 9 9
Chennai 7 9
Kolkata 9 4
PS B:\sem4\23bcp153_daa\lab4>
```

Analysis:

Operation	Array	Linked List
Insertion	$O(1)$	$O(1)$
Deletion	$O(n)$	$O(n)$
Search	$O(n)$	$O(n)$
Print	$O(n)$	$O(n)$

Advantages of Array based implementation:

- Accessing element takes constant time
- No extra pointer needed for traversal

Disadvantages of Array based implementation:

- Resizing takes extra time
- Even after dynamic resizing – this particular implementation may have space unused

Advantages of Linked List based implementation:

- Has proper dynamic size
- Insertion is efficient as we insert at head

Disadvantages of Linked List based implementation:

- Accessing elements/ traversal has o(n) complexity

Yes, sorting would help in array-based implementation as operations can use binary search

Except insertion which would require shifting

Yes, alphabetical order may slow down the operations as shifting would be required in array based implementation