VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

DATA STRUCTURES AND ALGORITHMS (BCSE202P)

LAB ASSESSMENT 2

Student Name: Yashvi Goyal

Student ID: 22BCE3019

Course Title: Data Structures and Algorithms

Experiment 1

Aim: Write a program in C to implement the following data structures: Singly linked list, doubly linked list, singly linked circular list, doubly linked circular list. Perform insertion and deletion on these data structures.

Code for insertion in a circular linked list:

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

struct node {
    int info;
    struct node* next;
};


struct node* last = NULL;

{
    int data;

    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));

    printf("\nEnter data to be inserted : \n");
    scanf("%d", &data);

    if (last == NULL) {
        temp->info = data;
        temp->next = temp;
        last = temp;
    }


    else {
        temp->info = data;
        temp->next = last->next;
        last->next = temp;
        last = temp;
    }
}

{
    int data, value;

     struct node *temp, *n;
```

```
    printf("\nEnter number after which"
           " you want to enter number: \n");
    scanf("%d", &value);
    temp = last->next;

    do {

            n = (struct node*)malloc(sizeof(struct node));
            printf("\nEnter data to be"
                   " inserted : \n");
            scanf("%d", &data);
            n->info = data;
            n->next = temp->next;
            temp->next = n;
            if (temp == last)
                last = n;
            break;
        }
        else
            temp = temp->next;
    } while (temp != last->next);
}

void viewList()
{
    if (last == NULL)
        printf("\nList is empty\n");

        struct node* temp;
        temp = last->next;
        do {
            printf("\nData = %d", temp->info);
            temp = temp->next;
        } while (temp != last->next);
    }
}

{
    addatlast();
    addatlast();
    addatlast();
    insertafter();


    viewList();

    return 0;
}
```

Output:

```
Enter data to be inserted :
10

Enter data to be inserted :
20

Enter data to be inserted :
30

Enter number after which you want to enter number :
10

Enter data to be inserted :
15

Data = 10
Data = 15
Data = 20
Data = 30
```

Code for deletiontion in a circular linked list:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int info;
    struct node* next;
};
void addatlast(int data)
{
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));

    if (last == NULL) {
        temp->info = data;
        temp->next = temp;
        last = temp;
    }
    else {
        temp->info = data;
        temp->next = last->next;
        last->next = temp;
        last = temp;
    }
}
void deletefirst()
{
    struct node* temp;

    if (last == NULL)
        printf("\nList is empty.\n");

    else {
        temp = last->next;
        last->next = temp->next;
        free(temp);
    }
}
void viewList()
{
    if (last == NULL)
        printf("\nList is empty\n");

    else {
        struct node* temp;
        temp = last->next;
        do {
            printf("\nData = %d", temp->info);
            temp = temp->next;
        } while (temp != last->next);
    }
```

```
}
int main()
{
    addatlast(10);
    addatlast(20);
    addatlast(30);

    printf("Before deletion:\n");
    viewList();

    deletefirst();

    printf("\n\nAfter deletion:\n");
    viewList();

    return 0;
}
```

Output:

```
Before deletion:

Data = 10
Data = 20
Data = 30

After deletion:

Data = 20
Data = 30
```

Experiment 2

Aim: Write a program in C to solve Towers of Hanoi problem

Code:

```c
#include <stdio.h>

// C recursive function to solve tower of hanoi puzzle
void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod)
{
    if (n == 1)
    {
        printf("\n Move disk 1 from rod %c to rod %c", from_rod, to_rod);
        return;
    }
    towerOfHanoi(n-1, from_rod, aux_rod, to_rod);
    printf("\n Move disk %d from rod %c to rod %c", n, from_rod, to_rod);
    towerOfHanoi(n-1, aux_rod, to_rod, from_rod);
}

int main()
{
    int n = 4; // Number of disks
    towerOfHanoi(n, 'A', 'C', 'B');  // A, B and C are names of rods
    return 0;
}
```

```
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 3 from rod A to rod B
Move disk 1 from rod C to rod A
Move disk 2 from rod C to rod B
Move disk 1 from rod A to rod B
Move disk 4 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 2 from rod B to rod A
Move disk 1 from rod C to rod A
Move disk 3 from rod B to rod C
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
```

Experiment 3

Aim:

To design and write an algorithm and program to perform insertion sort.

CODE

```c
#include <stdio.h>

int arr[] = {10, 6, 1, 5, 3};
int n = sizeof(arr)/sizeof(arr[0]), temp, i, j;

void insertionsort();
void display();

int main(){
    display();
    insertionsort();
    return 0;
}
```

```
void insertionsort(){
    for (i = 1; i<n; i++){
        temp = arr[i];
        j = i-1;

        while (j>=0 && temp<= arr[j]){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = temp;
        display();
    }

}

void display(){
    for(int i = 0; i<n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

OUTPUT

```
10 6 1 5 3
6 10 1 5 3
1 6 10 5 3
1 5 6 10 3
1 3 5 6 10
```

Experiment 4

Aim:

To design and write an algorithm and program to perform selection sort.

CODE

```c
#include <stdio.h>

int arr[] = {20, 12, 5, 18, 3};
int n = sizeof(arr)/sizeof(arr[0]), i, j, temp, min;

void selectionsort();
void display();

int main(){
    display();
    selectionsort();
    return 0;
}

void selectionsort(){
    for (i=0; i<n; i++){
        min = i;
        for (j = i; j<n; j++){
            if (arr[j] < arr[i]){
                min = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;

        display();
    }
}

void display(){
    for(int k = 0; k<n; k++){
        printf("%d ", arr[k]);
    }
    printf("\n");
}
```

OUTPUT

```
20 12 5 18 3
3 12 5 18 20
3 5 12 18 20
3 5 12 18 20
3 5 12 18 20
3 5 12 18 20
```

Experiment 5

Aim:

To design and write an algorithm and program to perform bubble sort.

CODE

```c
#include <stdio.h>

int array[] = {20, 9, 6, 3, 1};
int n = sizeof(array)/sizeof(array[0]), temp, i, j;

void bubblesort();
void display();

int main(){
    display();
    bubblesort();
    return 0;
}
void bubblesort(){
    for (i=0;i<n;i++){
        for (j=0; j<n-i-1; j++){
            if (array[j] > array[j+1]){
                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
            display();
        }
    }
}
void display(){
    for (int i=0; i<n; i++){
        printf("%d ", array[i]);
    }
    printf("\n");
}
```

OUTPUT

```
20 9 6 3 1
9 20 6 3 1
9 6 20 3 1
9 6 3 20 1
9 6 3 1 20
6 9 3 1 20
6 3 9 1 20
6 3 1 9 20
3 6 1 9 20
3 1 6 9 20
1 3 6 9 20
```

Experiment 6

Aim:

To design and write an algorithm and program to perform counting sort.

CODE

```c
#include <stdio.h>
#define MAX_SIZE 10
#define MAX_VALUE 10

int arr[] = {4, 2, 2, 8, 3, 3, 1};
int n = sizeof(arr)/sizeof(arr[0]), i, j, co;
int count[MAX_VALUE];
int output[MAX_SIZE];

void display(int array[], int size);
void countsort();

int main(){
    countsort();

    return 0;
}

void countsort(){
    printf("Initial array is: \n");
    display(arr, n);

    // Find max element in array
    int max = arr[0];
    for(i = 1; i<n; i++)
        if (arr[i]>max) max = arr[i];

    // Initialise array of max size
    int count[MAX_VALUE+1];
    for(i=0;i<max+1;i++)
        count[i] = 0;

    // Find the count of each element and store in array
    for (i=0; i<n; i++)
        count[arr[i]]++;
    printf("Count of each element in array is: \n");
    display(count, max+1);
    printf("\n");

    // Find the cumulative sum
    for (i=1; i<max+1; i++)
        count[i] += count[i-1];
    printf("Cumulative sum array is: \n");
```

```
    display(count, max+1);
    printf("\n");

    //
    for (i=0; i<n; i++){
        output[count[arr[i]] - 1] = arr[i];
        count[arr[i]]--;
        display(output, n);
        display(count, max+1);
        printf("\n");
    }

    printf("Sorted array is: \n");
    display(output, n);

}

void display(int array[], int size){
    for(int i = 0; i<size; i++){
        printf("%d ", array[i]);
    }
    printf("\n");
}
```

OUTPUT

```
Initial array is:
4 2 2 8 3 3 1
Count of each element in array is:
0 1 2 2 1 0 0 0 1

Cumulative sum array is:
0 1 3 5 6 6 6 6 7

0 0 0 0 0 4 0
0 1 3 5 5 6 6 6 7

0 0 2 0 0 4 0
0 1 2 5 5 6 6 6 7

0 2 2 0 0 4 0
0 1 1 5 5 6 6 6 7

0 2 2 0 0 4 8
0 1 1 5 5 6 6 6 6

0 2 2 0 3 4 8
0 1 1 4 5 6 6 6 6

0 2 2 3 3 4 8
0 1 1 3 5 6 6 6 6

1 2 2 3 3 4 8
0 0 1 3 5 6 6 6 6

Sorted array is:
1 2 2 3 3 4 8
```