**Data Structures (Lab)**

**Assignment- 5**

**(a) Creation of Circular linked list**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    printLinkedList(head);

    return 0;

}

**(b) Insertion at beginning**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

struct Node\* insertFirst(struct Node\* head, int data) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->next = head;

    struct Node\* temp = head;

    while (temp->next != head) {

        temp = temp->next;

    }

    temp->next = newNode;

    head = newNode;

    return head;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    head = insertFirst(head, 44);

    printLinkedList(head);

    return 0;

}

**(c) Insertion at remaining**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

struct Node\* insertFirst(struct Node\* head, int data) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->next = head;

    struct Node\* temp = head;

    while (temp->next != head) {

        temp = temp->next;

    }

    temp->next = newNode;

    head = newNode;

    return head;

}

struct Node\* insert(struct Node\* head, int index, int data) {

    if (index == 0) {

        return insertFirst(head, data);

    }

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->next = NULL;

    struct Node\* temp = head;

    for (int i = 0; i < index - 1; i++) {

        temp = temp->next;

    }

    newNode->next = temp->next;

    temp->next = newNode;

    return head;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    head = insert(head, 1, 44);

    printLinkedList(head);

    return 0;

}

**(d) Deletion at beginning**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

struct Node\* deleteFirst(struct Node\* head) {

    if (head == NULL) {

        return NULL;

    }

    struct Node\* temp = head;

    while (temp->next != head) {

        temp = temp->next;

    }

    temp->next = head->next;

    return temp->next;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    head = deleteFirst(head);

    printLinkedList(head);

    return 0;

}

**(e) traverse**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    printLinkedList(head);

    return 0;

}

**(f) Search**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

int search(struct Node\* head, int data) {

    struct Node\* temp = head;

    do {

        if (temp->data == data) {

            return 1;

        }

        temp = temp->next;

    } while (temp != head);

    return 0;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    cout << search(head, 55);

    return 0;

}

**(g) sort**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

void swap(struct Node\* a, struct Node\* b) {

    int temp = a->data;

    a->data = b->data;

    b->data = temp;

}

struct Node\* sort(struct Node\* head) {

    if (head == NULL || head->next == head) return head;

    struct Node\* end = NULL;

    int swapped;

    do {

        swapped = 0;

        struct Node\* current = head;

        while (current->next != end && current->next != head) {

            if (current->data > current->next->data) {

                swap(current, current->next);

                swapped = 1;

            }

            current = current->next;

        }

        end = current;

    } while (swapped);

    return head;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    head = sort(head);

    printLinkedList(head);

    return 0;

}

**(h) update**

#include <iostream>

using namespace std;

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* circularLinkedList(int arr[], int size) {

    struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

    head->data = arr[0];

    head->next = NULL;

    struct Node\* temp = head;

    for (int i = 1; i < size; i++) {

        struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

        newNode->data = arr[i];

        newNode->next = NULL;

        temp->next = newNode;

        temp = temp->next;

    }

    temp->next = head;

    return head;

}

struct Node\* update(struct Node\* head, int data, int index) {

    if (head == NULL) {

        return head;

    }

    struct Node\* temp = head;

    for (int i = 0; i < index - 1; i++) {

        temp = temp->next;

    }

    temp->data = data;

    return head;

}

void printLinkedList(struct Node\* head) {

    struct Node\* temp = head;

    do {

        cout << temp->data << " ";

        temp = temp->next;

    } while (temp != head);

}

int main(int argc, char const\* argv[]) {

    int arr[] = {1, 55, 2, 56, 1, 2, 3, 45};

    int n = sizeof(arr) / sizeof(arr[0]);

    struct Node\* head = circularLinkedList(arr, n);

    head = update(head, 100, 3);

    printLinkedList(head);

    return 0;

}