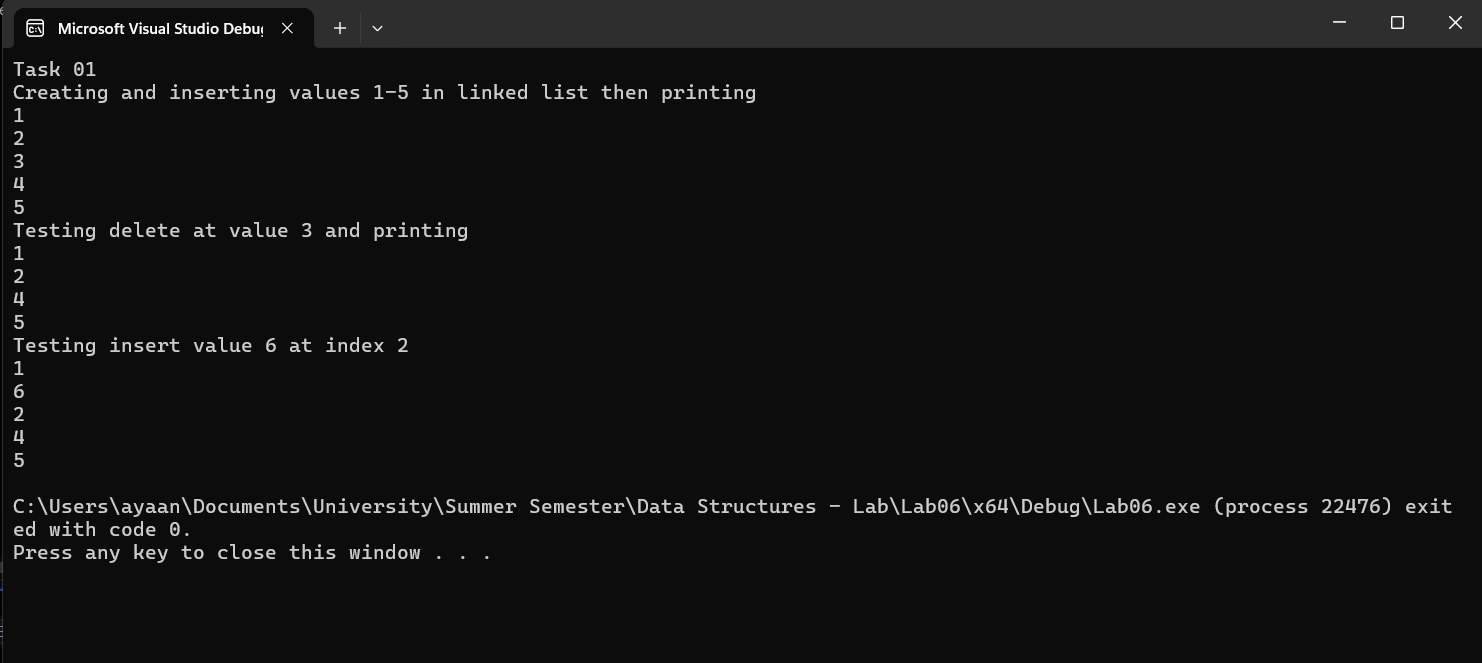
Q1:



#include<iostream>

using namespace std;

class Node

{

public:

int data;

Node\* next;

Node()

{

data = 0;

next = NULL;

}

Node(int x1, Node\* x3)

{

data = x1;

next = x3;

}

void setData(int x1)

{

data = x1;

}

void setNext(Node\* x1)

{

next = x1;

}

int getData()

{

return data;

}

Node\* getNext()

{

return next;

}

};

class CircularLinkedList

{

public:

Node\* head;

Node\* current;

CircularLinkedList()

{

head = NULL;

current = NULL;

}

Node\* getHead()

{

return head;

}

void insert(int x)

{

Node\* ptr = new Node;

ptr->setData(x);

if (head == NULL)

{

head = ptr;

ptr->setNext(head);

return;

}

current = head;

while (current->getNext() != head)

{

current = current->getNext();

}

current->setNext(ptr);

ptr->setNext(head);

}

bool isEmpty()

{

if (head == NULL)

return 1;

return 0;

}

int search(int x)

{

current = head;

int count = 0;

if (head == NULL)

{

return 0;

}

current = current->getNext();

while (current != head)

{

if (current->getData() == x)

return 1;

current = current->getNext();

count++;

}

return 0;

}

void update(int val1, int val2)

{

current = head;

current = current->getNext();

while (current != head)

{

if (current->getData() == val1)

{

current->setData(val2);

return;

}

current = current->getNext();

}

}

void insertAtIndex(int x, int idx)

{

Node\* ptr = new Node;

ptr->setData(x);

if (idx < 0)

{

return;

}

if (idx == 0)

{

if (head == NULL)

{

head = ptr;

ptr->setNext(head);

}

else

{

ptr->setNext(head);

current = head;

while (current->next != head)

{

current = current->next;

}

head = ptr;

current->next = head;

}

return;

}

int count = 1;

current = head;

// current = current->getNext();

if (head == NULL)

{

return;

}

do

{

if (count == idx)

{

ptr->setNext(current->getNext());

current->setNext(ptr);

return;

}

count++;

current = current->getNext();

} while (current != head);

}

void deleteData(int x)

{

Node\* previous = NULL;

current = head;

Node\* ptr = NULL;

ptr = head;

int count = 0;

do

{

if (current->getData() == x)

break;

previous = current;

current = current->getNext();

count++;

} while (current != head);

if (current == head)

{

while (ptr->getNext() != head)

{

ptr = ptr->getNext();

}

previous = ptr;

head = current->getNext();

previous->setNext(head);

delete current;

}

else if (current != head)

{

previous->setNext(current->getNext());

delete current;

}

}

void print()

{

current = head;

do

{

cout << current->data << endl;

current = current->next;

} while (current != head);

}

};

int main()

{

//Q1

cout << "Task 01\n";

cout << "Creating and inserting values 1-5 in linked list then printing\n";

CircularLinkedList list1;

list1.insert(1);

list1.insert(2);

list1.insert(3);

list1.insert(4);

list1.insert(5);

list1.print();

cout << "Testing delete at value 3 and printing\n";

list1.deleteData(3);

list1.print();

cout << "Testing insert value 6 at index 2\n";

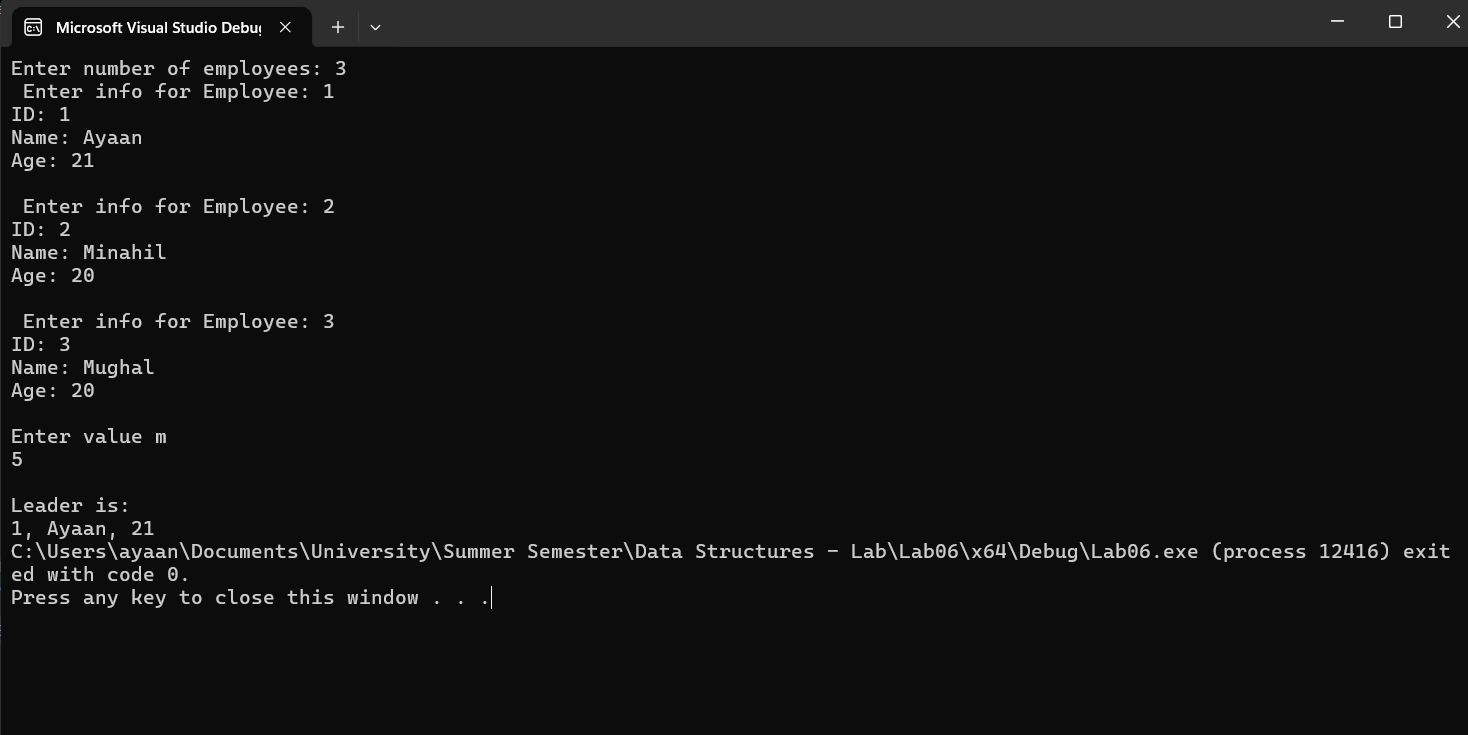
list1.insertAtIndex(6, 1);

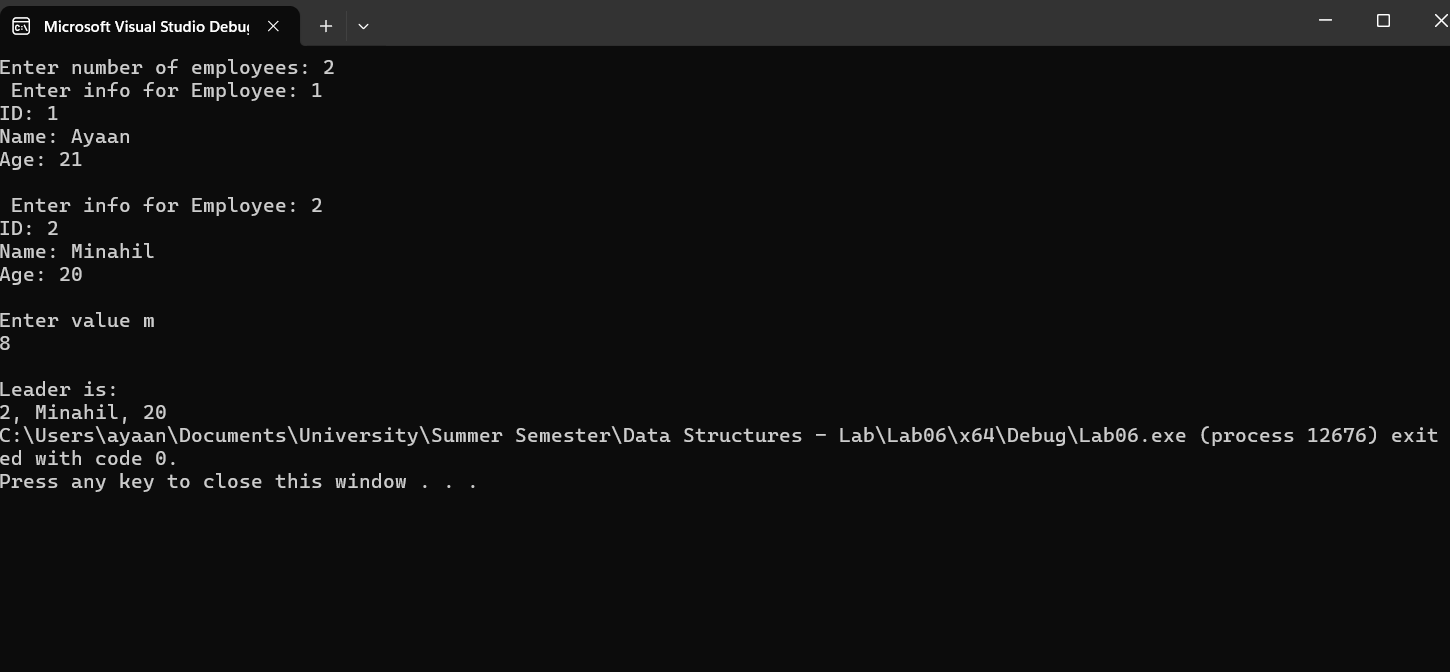
list1.print();

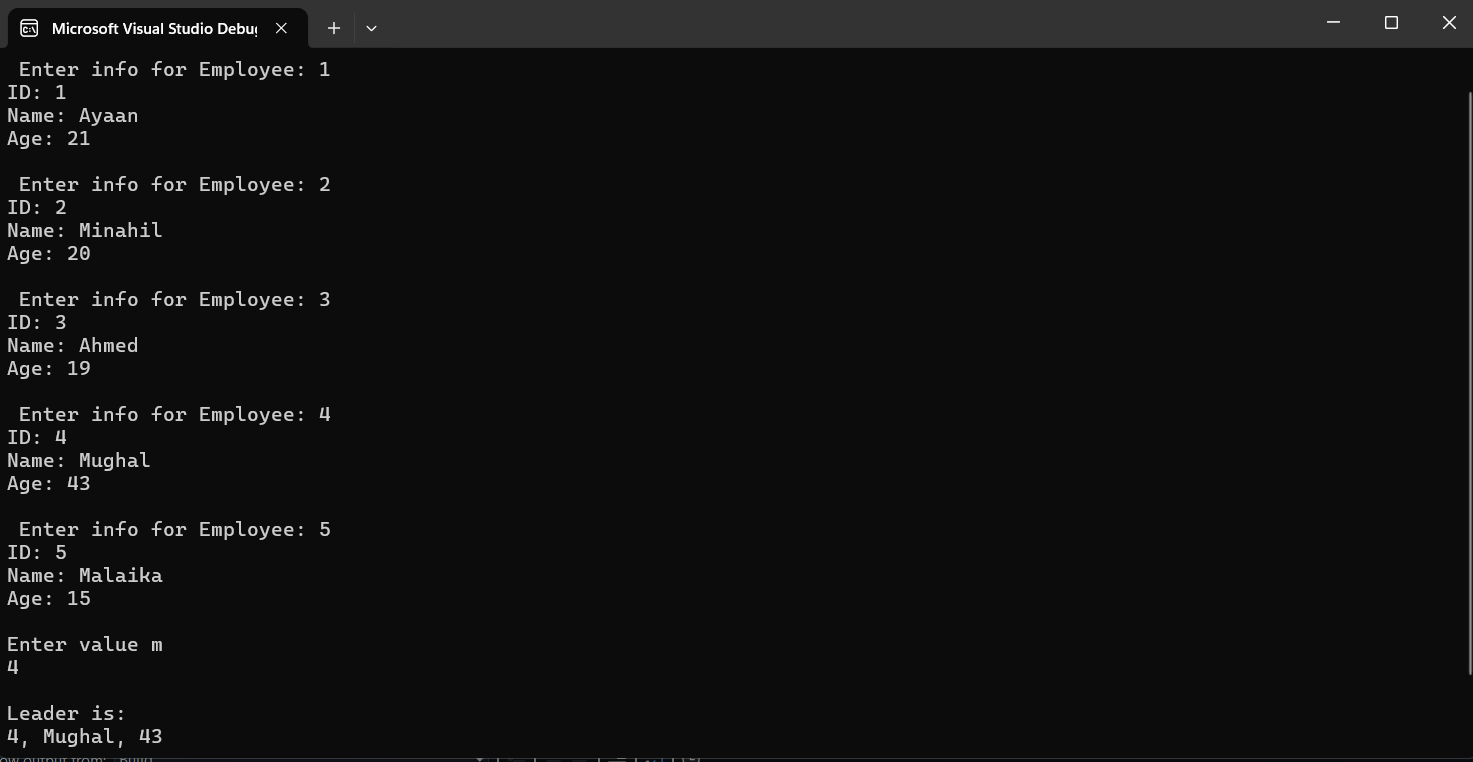
return 0;

}

Q2:







#include<iostream>

using namespace std;

#include<string>

struct Data

{

int ID;

string name;

int age;

};

class Node

{

public:

Data data;

Node\* next;

Node()

{

data = { 0, "", };

next = NULL;

}

Node(Data x1, Node\* x3)

{

data = x1;

next = x3;

}

void setData(Data x1)

{

data = x1;

}

void setNext(Node\* x1)

{

next = x1;

}

Data getData()

{

return data;

}

Node\* getNext()

{

return next;

}

};

class CircularLinkedList

{

public:

Node\* head;

Node\* current;

CircularLinkedList()

{

head = NULL;

current = NULL;

}

Node\* getHead()

{

return head;

}

void insert(Data x)

{

Node\* ptr = new Node;

ptr->setData(x);

if (head == NULL)

{

head = ptr;

ptr->setNext(head);

return;

}

current = head;

while (current->getNext() != head)

{

current = current->getNext();

}

current->setNext(ptr);

ptr->setNext(head);

}

bool isEmpty()

{

if (head == NULL)

return 1;

return 0;

}

int search(int x)

{

current = head;

int count = 0;

if (head == NULL)

{

return 0;

}

current = current->getNext();

while (current != head)

{

if (current->getData().ID == x)

return 1;

current = current->getNext();

count++;

}

return 0;

}

void update(Data val1, Data val2)

{

current = head;

current = current->getNext();

while (current != head)

{

if (current->getData().ID == val1.ID)

{

current->setData(val2);

return;

}

current = current->getNext();

}

}

void insertAtIndex(Data x, int idx)

{

Node\* ptr = new Node;

ptr->setData(x);

if (idx < 0)

{

return;

}

if (idx == 0)

{

if (head == NULL)

{

head = ptr;

ptr->setNext(head);

}

else

{

ptr->setNext(head);

current = head;

while (current->next != head)

{

current = current->next;

}

head = ptr;

current->next = head;

}

return;

}

int count = 1;

current = head;

// current = current->getNext();

if (head == NULL)

{

return;

}

do

{

if (count == idx)

{

ptr->setNext(current->getNext());

current->setNext(ptr);

return;

}

count++;

current = current->getNext();

} while (current != head);

}

void deleteData(int x)

{

Node\* previous = NULL;

current = head;

Node\* ptr = NULL;

ptr = head;

int count = 0;

do

{

if (current->getData().ID == x)

break;

previous = current;

current = current->getNext();

count++;

} while (current != head);

if (current == head)

{

while (ptr->getNext() != head)

{

ptr = ptr->getNext();

}

previous = ptr;

head = current->getNext();

previous->setNext(head);

delete current;

}

else if (current != head)

{

previous->setNext(current->getNext());

delete current;

}

}

void print()

{

current = head;

do

{

cout << current->data.ID <<", "<<current->data.name<<", "<<current->data.age << endl;

current = current->next;

} while (current != head);

}

int countNodes()

{

current = head;

if (head == NULL)

{

return 0;

}

int count = 0;

do {

count++;

current = current->next;

} while (current != head);

return count;

}

int getID(int x)

{

current = head;

for (int i = 1; i < x; i++)

{

current = current->next;

}

return current->data.ID;

}

Data findLeader(int m)

{

while (countNodes() != 1)

{

deleteData(getID(m + 1));

}

return head->data;

}

};

int main()

{

int x;

cout << "Enter number of employees: ";

cin >> x;

Data\* data = new Data[x];

CircularLinkedList list1;

for (int i = 0; i < x; i++)

{

cout << " Enter info for Employee: " << i + 1 << endl;

cout << "ID: ";

cin >> data[i].ID;

cout << "Name: ";

cin >> data[i].name;

cout << "Age: ";

cin >> data[i].age;

cout << endl;

list1.insert(data[i]);

}

int m;

cout << "Enter value m\n";

cin >> m;

cout << endl;

Data winner;

winner = list1.findLeader(m);

cout << "Leader is: \n";

cout << winner.ID << ", " << winner.name << ", " << winner.age;

}