**Module 4**

#include <iostream>

using namespace std; **Topic 2**

class node

{

public:

  int data;

  node \*next;

};

class linked\_list

{

private:

  node \*head, \*tail;

  int n;

public:

  linked\_list()

  {

    head = NULL;

    tail = NULL;

  }

  void create\_node()

  {

    int x;

    cout << "How many nodes you need? = ";

    cin >> n;

    cout << "ENter your node values" << endl;

    for (int i = 1; i <= n; i++)

    {

      cin >> x;

      node \*tmp = new node;

      tmp->data = x;

      tmp->next = NULL;

      if (head == NULL)

      {

        head = tmp;

      }

      if (tail == NULL)

      {

        tail = tmp;

      }

      else

      {

        tail->next = tmp;

        tail = tmp;

      }

    }

  }

  void enter\_node()

  {

    bool g;

    int a, b, c;

    node \*p = head;

    node \*q = new node;

    node \*prev = new node;

    cout << "What do you want\n1.before\n2.After?" << endl;

    cin >> c;

    if (c == 2)

    {

      cout << "Enter the node value and to be inserted node value" << endl;

      cin >> a >> b;

      if (p == NULL)

      {

        cout << "List empty" << endl;

      }

      while (p != NULL)

      {

        g = true;

        if (p->data == a)

        {

          q->data = b;

          q->next = p->next;

          p->next = q;

          break;

        }

        else

        {

          g = false;

        }

        p = p->next;

      }

    }

    else if (c == 1)

    {

      cout << "Enter the node value and to be inserted node value" << endl;

      cin >> a >> b;

      if (p == NULL)

      {

        cout << "List empty" << endl;

      }

      while (p != NULL)

      {

        g = true;

        if (p == head && p->data == a)

        {

          q->data = b;

          q->next = p;

          head = q;

          break;

        }

        else if (p->data == a)

        {

          q->data = b;

          prev->next = q;

          q->next = p;

          break;

        }

        else

        {

          g = false;

        }

        prev = p;

        p = p->next;

      }

    }

    if (g == false)

    {

      cout << "Your node value is not in the list" << endl;

    }

  }

  void delete\_node()

  {

    bool h;

    int b;

    cout << "Enter the node value to be deleted" << endl;

    cin >> b;

    node \*p = head;

    node \*prev = NULL;

    if (p == NULL)

    {

      cout << "List Empty" << endl;

    }

    while (p != NULL)

    {

      h = true;

      if (p->data == b)

      {

        if (p == head)

        {

          head = p->next;

          break;

        }

        else

        {

          prev->next = p->next;

          break;

        }

      }

      else

      {

        h = false;

      }

      prev = p;

      p = p->next;

    }

    if (h == false)

    {

      cout << "Your node value is not in the list" << endl;

    }

  }

  void update\_node()

  {

    bool i;

    int a, b;

    cout << "Enter the node value and new value" << endl;

    cin >> b >> a;

    node \*p = head;

    // node \*prev = NULL;

    if (p == NULL)

    {

      cout << "List Empty" << endl;

    }

    while (p != NULL)

    {

      i = true;

      if (p->data == b)

      {

        p->data = a;

        break;

      }

      else

      {

        p = p->next;

      }

      i = false;

    }

    if (i == false)

    {

      cout << "Your node value is not in the list" << endl;

    }

  }

  void Display()

  {

    node \*temp = head;

    if (head == NULL)

    {

      cout << "Can't Display nodes" << endl;

    }

    // cout << "h";

    while (temp != NULL)

    {

      cout << temp->data << " ";

      temp = temp->next;

    }

    cout << endl;

  }

  void get\_ad()

  {

    int i = 0;

    node \*temp = head;

    if (head == NULL)

    {

      cout << "List empty" << endl;

    }

    while (temp != NULL)

    {

      cout << i << " " << temp << " " << temp->data << " " << temp->next << endl;

      temp = temp->next;

      i++;

    }

    cout << endl;

  }

};

void menu()

{

  cout << endl;

  cout << "\*\*\*\*\*\*\*\* Menu \*\*\*\*\*" << endl;

  cout << "1. Create \n2. Insert \n3. Delete \n4. Update \n5. Display \n6. Exit" << endl;

  cout << "Enter your option: ";

}

int main()

{

  linked\_list a;

  int m;

  while (m != 6)

  {

    menu();

    cin >> m;

    switch (m)

    {

    case 1:

      a.create\_node();

      a.Display();

      break;

    case 2:

      a.enter\_node();

      a.Display();

      break;

    case 3:

      a.delete\_node();

      a.Display();

      break;

    case 4:

      a.update\_node();

      a.Display();

      break;

    case 5:

      a.get\_ad();

      break;

    case 6:

      break;

    }

  }

  return 0;}

#include <iostream> **Topic 3**

using namespace std;

class node

{

public:

    int data;

    node \*next;

};

class linked\_list

{

private:

    node \*head, \*tail;

    int n;

public:

    linked\_list()

    {

        head = NULL;

        tail = NULL;

    }

    void create\_node()

    {

        int x;

        cout << "How many nodes you need?" << endl;

        cin >> n;

        cout << "ENter your node values" << endl;

        for (int i = 1; i <= n; i++)

        {

            cin >> x;

            node \*tmp = new node;

            tmp->data = x;

            tmp->next = NULL;

            if (head == NULL)

            {

                head = tmp;

            }

            if (tail == NULL)

            {

                tail = tmp;

            }

            else

            {

                tail->next = tmp;

                tail = tmp;

            }

        }

        tail->next = head;

    }

    void Display()

    {

        node \*temp = head;

        if (head == NULL)

        {

            cout << "Can't Display Nodes" << endl;

        }

        // cout << "h";

        while (temp != NULL)

        {

            // cout << "--> " << temp->data << " ";

            cout << temp->data << " ";

            // cout << i << " " << temp << " " << temp->data << " " << temp->next << endl;

            temp = temp->next;

            if (temp == head)

                break;

        }

    }

    void enter\_node()

    {

        bool g;

        int a, b, c;

        node \*p = head;

        node \*q = new node;

        node \*prev = new node;

        cout << "What do you want\n1.before\n2.After?" << endl;

        cin >> c;

        if (c == 2)

        {

            cout << "Enter the node value and to be inserted node value" << endl;

            cin >> a >> b;

            if (p == NULL)

            {

                cout << "List Empty" << endl;

            }

            while (p != NULL)

            {

                g = true;

                if (p->data == a)

                {

                    q->data = b;

                    q->next = p->next;

                    p->next = q;

                    break;

                }

                else

                {

                    g = false;

                }

                p = p->next;

                if (p == head)

                    break;

            }

            if (g == false)

            {

                cout << "Your node value is not in the list" << endl;

            }

        }

        else if (c == 1)

        {

            node \*head2 = NULL;

            cout << "Enter the node value and to be inserted node value" << endl;

            cin >> a >> b;

            if (p == NULL)

            {

                cout << "List Empty" << endl;

            }

            while (p != NULL)

            {

                g = true;

                if (p == head && p->data == a)

                {

                    q->data = b;

                    q->next = p;

                    head2 = head;

                    head = q;

                    p = p->next;

                }

                else if (p->data == a)

                {

                    q->data = b;

                    prev->next = q;

                    q->next = p;

                    break;

                }

                else if (p->next == head2)

                {

                    p->next = head;

                    break;

                }

                else

                {

                    g = false;

                }

                prev = p;

                p = p->next;

                if (p == head)

                    break;

            }

            if (g == false && head2 == NULL)

            {

                cout << "Your node value is not in the list" << endl;

            }

        }

    }

    void update\_node()

    {

        bool i;

        int a, b;

        cout << "Enter the node value and new value" << endl;

        cin >> b >> a;

        node \*p = head;

        // node \*prev = NULL;

        if (p == NULL)

        {

            cout << "List Empty" << endl;

        }

        while (p != NULL)

        {

            i = true;

            if (p->data == b)

            {

                p->data = a;

                break;

            }

            else

            {

                i = false;

            }

            p = p->next;

            if (p == head)

            {

                break;

            }

        }

        if (i == false)

        {

            cout << "Your node value is not in the list" << endl;

        }

    }

    void delete\_node()

    {

        bool h;

        int a, b;

        cout << "Enter the node value to be deleted" << endl;

        cin >> b;

        node \*p = head;

        node \*prev = NULL;

        if (p == NULL)

        {

            cout << "List Empty" << endl;

        }

        while (p != NULL)

        {

            h = true;

            if (p->data == b)

            {

                if (p == head)

                {

                    head = p->next;

                    break;

                }

                else

                {

                    prev->next = p->next;

                    break;

                }

            }

            else

            {

                h = false;

            }

            prev = p;

            p = p->next;

            if (p == head)

                break;

        }

        if (h == false)

        {

            cout << "Your node value is not in the list" << endl;

        }

    }

    void get\_ad()

    {

        node \*temp = head;

        if (head == NULL)

        {

            cout << "List empty" << endl;

        }

        // cout << "h";

        int i = 1;

        cout << "no node        data next" << endl;

        while (temp != NULL)

        {

            cout << i << " " << temp << " " << temp->data << " " << temp->next << endl;

            temp = temp->next;

            i++;

            if (temp == head)

                break;

        }

        cout << endl;

    }

};

void menu()

{

    cout << endl;

    cout << "\*\*\*\*\*\*\*\* Menu \*\*\*\*\*" << endl;

    cout << "Circular Linked List" << endl;

    cout << "1. Create \n2. Insert \n3. Delete \n4. Update \n5. Display \n6. Exit" << endl;

    cout << "Enter your option: ";

}

int main()

{

    linked\_list a;

    int m;

    while (m != 6)

    {

        menu();

        cin >> m;

        switch (m)

        {

        case 1:

            a.create\_node();

            a.Display();

            break;

        case 2:

            a.enter\_node();

            a.Display();

            break;

        case 3:

            a.delete\_node();

            a.Display();

            break;

        case 4:

            a.update\_node();

            a.Display();

            break;

        case 5:

            a.get\_ad();

            break;

        case 6:

            break;

        }

    }

    return 0;

}

#include <iostream>

using namespace std;

class node **Topic 4**

{

public:

    int data;

    node \*next;

    node \*previ;

};

class linked\_list

{

private:

    node \*head, \*tail;

    int n;

public:

    linked\_list()

    {

        head = NULL;

        tail = NULL;

    }

    void create\_node()

    {

        int x;

        cout << "How many nodes you need?" << endl;

        cin >> n;

        cout << "ENter your node values" << endl;

        for (int i = 1; i <= n; i++)

        {

            cin >> x;

            node \*tmp = new node;

            tmp->data = x;

            tmp->previ = NULL;

            tmp->next = NULL;

            if (head == NULL)

            {

                head = tmp;

            }

            if (tail == NULL)

            {

                tail = tmp;

            }

            else

            {

                tail->next = tmp;

                tmp->previ = tail;

                tail = tmp;

            }

        }

    }

    void Display()

    {

        node \*temp = head;

        if (head == NULL)

        {

            cout << "Can't Display Nodes" << endl;

        }

        // cout << "h";

        while (temp != NULL)

        {

            // cout << "--> " << temp->data << " ";

            cout << temp->data << " ";

            temp = temp->next;

        }

        }

    void enter\_node()

    {

        bool g;

        int a, b, c;

        node \*p = head;

        node \*q = new node;

        cout << "What do you want\n1.before\n2.After?" << endl;

        cin >> c;

        if (c == 2)

        {

            cout << "Enter the node value and to be inserted node value" << endl;

            cin >> a >> b;

            if (p == NULL)

            {

                cout << "List empty" << endl;

            }

            while (p != NULL)

            {

                g = true;

                if (p->data == a)

                {

                    q->data = b;

                    q->next = p->next;

                    q->previ = p;

                    p->next = q;

                    break;

                }

                else

                {

                    g = false;

                }

                p = p->next;

            }

        }

        else if (c == 1)

        {

            cout << "Enter the node value and to be inserted node value" << endl;

            cin >> a >> b;

            if (p == NULL)

            {

                cout << "List empty" << endl;

            }

            while (p != NULL)

            {

                g = true;

                if (p == head && p->data == a)

                {

                    q->data = b;

                    q->next = head;

                    q->previ = NULL;

                    head->previ = q;

                    head = q;

                    break;

                }

                else if (p->data == a)

                {

                    q->data = b;

                    q->next = p;

                    q->previ = p->previ;

                    p->previ->next = q;

                    p->previ = q;

                    break;

                }

                else

                {

                    g = false;

                }

                p = p->next;

            }

        }

        if (g == false)

        {

            cout << "Your node value is not in the list" << endl;

        }

    }

    void update\_node()

    {

        bool i;

        int a, b;

        cout << "Enter the node value and new value" << endl;

        cin >> b >> a;

        node \*p = head;

        // node \*prev = NULL;

        if (p == NULL)

        {

            cout << "List Empty" << endl;

        }

        while (p != NULL)

        {

            i = true;

            if (p->data == b)

            {

                p->data = a;

                break;

            }

            else

            {

                i = false;

            }

            p = p->next;

            if (p == head)

            {

                break;

            }

        }

        if (i == false)

        {

            cout << "Your node value is not in the list" << endl;

        }

    }

    void delete\_node()

    {

        bool h;

        int a, b;

        cout << "Enter the node value to be deleted" << endl;

        cin >> b;

        node \*p = head;

        if (p == NULL)

        {

            cout << "List Empty" << endl;

        }

        while (p != NULL)

        {

            h = true;

            if (p->data == b)

            {

                if (p == head)

                {

                    head = p->next;

                    p->next->previ = NULL;

                    break;

                }

                else

                {

                    p->previ->next = p->next;

                    break;

                }

            }

            else

            {

                h = false;

            }

            p = p->next;

            if (p == head)

            {

                break;

            }

        }

        if (h == false)

        {

            cout << "Your node value is not in the list" << endl;

        }

    }

    void get\_ad()

    {

        node \*temp = head;

        if (head == NULL)

        {

            cout << "List empty" << endl;

        }

        // cout << "h";

        int i = 1;

        cout << "no prev        node        data next" << endl;

        while (temp != NULL)

        {

            cout << i << " " << temp->previ << " " << temp << " " << temp->data << " " << temp->next << endl;

            temp = temp->next;

            i++;

        }

        cout << endl;

    }

};

void menu()

{

    cout << endl;

    cout << "\*\*\*\*\*\*\*\* Menu \*\*\*\*\*" << endl;

    cout << "Doubly Linked List" << endl;

    cout << "1. Create \n2. Insert \n3. Delete \n4. Update \n5. Display \n6. Exit" << endl;

    cout << "Enter your option: " << endl;

}

int main()

{

    linked\_list a;

    int m;

    while (m != 6)

    {

        menu();

        cin >> m;

        switch (m)

        {

        case 1:

            a.create\_node();

            a.Display();

            break;

        case 2:

            a.enter\_node();

            a.Display();

            break;

        case 3:

            a.delete\_node();

            a.Display();

            break;

        case 4:

            a.update\_node();

            a.Display();

            break;

        case 5:

            a.get\_ad();

            break;

        case 6:

            break;

        }

    }

    return 0;

}