DSA Lab05

23K2001

M.Muzammil Siddiqui

BCS-3J

Q1:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        node\* getHead(){ return head; }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == nullptr){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

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

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            node\* before = nullptr;

            node\* temp = head;

            while(temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

            }

            before->setNext(temp->getNext());

            delete temp;

        }

};

void displayReverse(node\* head){

    if(head==nullptr)

    return;

    displayReverse(head->getNext());

    cout<<head->getData()<<"\t";

}

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e,v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

    flex.display();

    cout<<"Insert an element at end: ";

    cin>>v;

    flex.insertAtEnd(v);

    cout<<"Displaying the list reverse by recursion:"<<endl;

    displayReverse(flex.getHead());

    return 0;

}

A screen shot of a computer

Description automatically generated

Q2:

//23K2001 - Muzammil

#include<iostream>

#include<stdlib.h>

#include<time.h>

using namespace std;

void evaluateGuess(int key,int p=0){

    int guess;

    cout<<"Enter guess (Player#"<<p<<"): ";

    cin>>guess;

    if(guess==key){

        cout<<"Player#"<<p<<" had the correct guess. ("<<guess<<")"<<endl;

        return;

    }

    else{

        if(guess>key) cout<<"Too high!"<<endl;

        else cout<<"Too low!"<<endl;

        evaluateGuess(key,1-p);

    }

}

int main(){

    srand(time(0));

    int k = (rand()%100)+1;

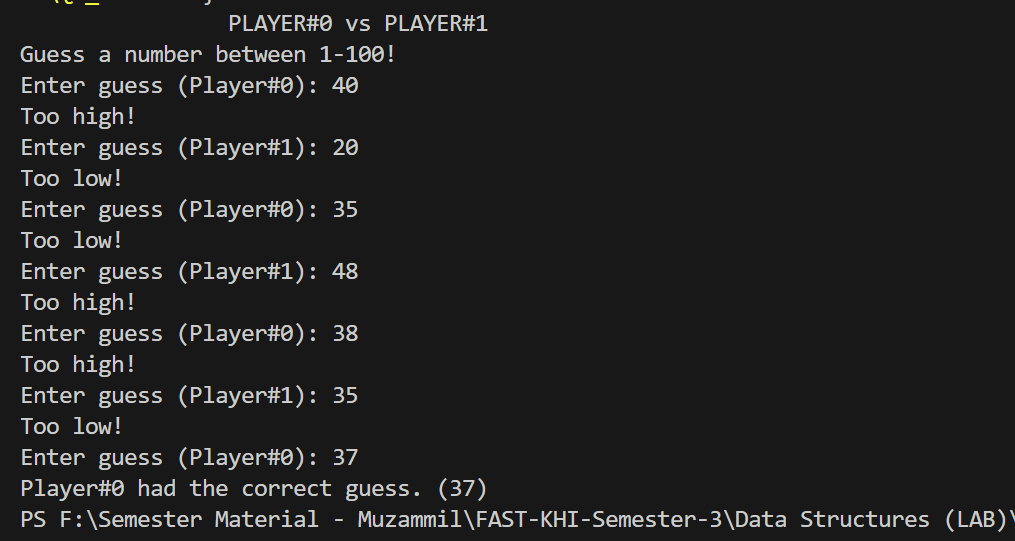
    cout<<"\t\tPLAYER#0 vs PLAYER#1"<<endl;

    cout<<"Guess a number between 1-100!"<<endl;

    evaluateGuess(k);

    return 0;

}



Q3:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        node\* getHead(){ return head; }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == nullptr){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

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

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            node\* before = nullptr;

            node\* temp = head;

            while(temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

            }

            before->setNext(temp->getNext());

            delete temp;

        }

};

int length(node\* head,int c=0){

    if(head==nullptr)

    return c;

    return length(head->getNext(),c+1);

}

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e,v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

    flex.display();

    cout<<"Length of list by tail recursion: ";

    cout<<length(flex.getHead());

    return 0;

}

A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

Q4:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        node\* getHead(){ return head; }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == nullptr){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

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

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            node\* before = nullptr;

            node\* temp = head;

            while(temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

            }

            before->setNext(temp->getNext());

            delete temp;

        }

};

bool search(node\* head,int key){

    if(head==nullptr)

    return false;

    if(head->getData()==key)

        return true;

    else

        search(head->getNext(),key);

}

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e,v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

    flex.display();

    cout<<"Enter value to search: ";

    cin>>v;

    if(search(flex.getHead(),v))

        cout<<"Value is present."<<endl;

    else

        cout<<"Value is NOT present."<<endl;

    return 0;

}

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

Q5:

//23K2001 - Muzammil

#include<iostream>

using namespace std;

int recursiveArraySum(int \*arr[],int \*sizes,int dim){

    if(dim==0)

        return 0;

    if(sizes[dim-1]==0)

        return recursiveArraySum(arr,sizes,dim-1);

    return (arr[dim-1][--sizes[dim-1]])+recursiveArraySum(arr,sizes,dim);

}

int main(){

    int \*\*flex;

    flex = new int\*[4];

    flex[0] = new int[2];

    flex[1] = new int[3];

    flex[2] = new int[4];

    flex[3] = new int[1];

    for(int i=0;i<2;i++)

    flex[0][i] = 3;

    for(int i=0;i<3;i++)

    flex[1][i] = 2;

    for(int i=0;i<4;i++)

    flex[2][i] = 4;

    for(int i=0;i<1;i++)

    flex[3][i] = 8;

    int sizes[] = {2,3,4,1};

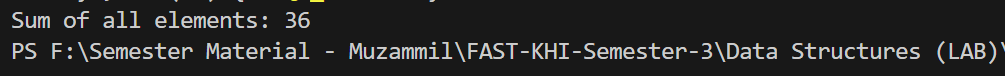
    cout<<"Sum of all elements: "<<recursiveArraySum(flex,sizes,4);

    for(int i=0;i<4;i++)

        delete[] flex[i];

    delete[] flex;

    return 0;

}

Q6:

//23K2001 - Muzammil

#include<iostream>

using namespace std;

bool moveLion(bool \*maze[],int x,int y,int s,bool \*\*sol){

    if(x==s-1 && y==s-1){

        sol[x][y]=1;

        return true;

    }

    if((x<s && y<s && maze[x][y]==1) ? true : false){

        sol[x][y]=1;

        if(moveLion(maze,x+1,y,s,sol))

            return true;

        if(moveLion(maze, x, y+1, s, sol))

            return true;

    }

    sol[x][y]=0;

    return false;

}

void display(bool \*\*arr,int r,int c){

    for(int i=0;i<r;i++){

        cout<<"-";

        for(int j=0;j<c;j++)

            cout<<arr[i][j]<<"-";

        cout<<endl;

    }

}

int main(){

    bool \*\*map, \*\*path;

    int m,n;

    cout<<"Enter map dimensions: ";

    cin>>m>>n;

    map = new bool\*[m];

    path = new bool\*[m];

    for(int i=0;i<m;i++){

        map[i] = new bool[n];

        path[i] = new bool[n];

    }

    cout<<"Create the maze:"<<endl;

    for(int i=0;i<m;i++){

        for(int j=0;j<n;j++){

            cin>>map[i][j];

            path[i][j]=false;

        }

    }

    cout<<endl<<"Maze:"<<endl;

    display(map,m,n);

    if(moveLion(map,0,0,5,path)){

        cout<<endl<<"Path found!"<<endl;

        cout<<endl<<"Movable path:"<<endl;

        display(path,m,n);

    }

    else

        cout<<"No path could be found!"<<endl;

    for(int i=0;i<m;i++){

        delete[] map[i];

        delete[] path[i];

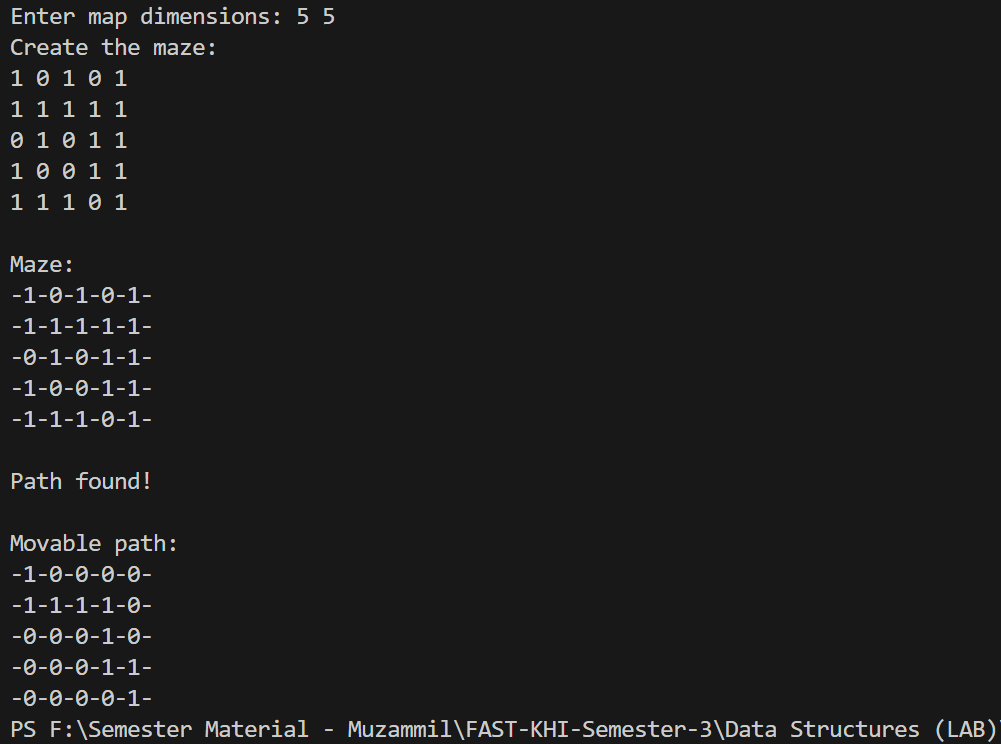
    }

    delete[] map;

    delete[] path;

    return 0;

}



Q7:

//23K2001 - Muzammil

#include<iostream>

using namespace std;

bool canPlace(int board[],int row,int col){

    for(int i=0; i<row; i++){

        if(board[i]==col || abs(board[i]-col)==abs(i-row))

            return false;

    }

    return true;

}

bool maxFlags(int board[],int n,int row=0){

    if(row==n)

        return true;

    for(int col=0;col<n;col++){

        if(canPlace(board,row,col)){

            board[row] = col;

            if(maxFlags(board,n,row+1))

                return true;

            board[row] = -1;

        }

    }

    return false;

}

void display(int \*arr,int n){

    int f=0;

    for(int i=0;i<n;i++){

        for(int j=0;j<n;j++){

            if(arr[i]==j){

                cout<<"F ";

                f++; }

            else

                cout<<"- ";

        }

        cout<<endl;

    }

    cout<<"Maximum number of flags that can be placed: "<<f<<endl;

}

int main(){

    int\* ground=new int[4];

    for(int i=0;i<4;i++)

        ground[i] = -1;

    maxFlags(ground,4);

    display(ground,4);

    delete[] ground;

    return 0;

}

