DSA Lab07

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;

        int size;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

            size = 0;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

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

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

            size++;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

            size++;

        }

        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);

            size++;

        }

        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;

            size--;

        }

        void reverseMN(int m,int n){

            if(head==nullptr || m < 1 || n > size || m >= n){

                cout<<"Error applying function!"<<endl;

                return;

            }

            node\* current = head;

            node\* before = nullptr;

            node\* newTail = nullptr;

            for(int i = 1;i<m;i++){

                before = current;

                current = current->getNext();

            }

            newTail = current;

            node\* next = nullptr;

            for(int i = m;i<=n;i++){

                next = current->getNext();

                current->setNext(before);

                before = current;

                current = next;

            }

            if(before!=nullptr){

                if(m == 1)

                    head = before;

                else

                {

                    node\* temp = head;

                    for (int i = 1; i < m - 1; i++)

                        temp = temp->getNext();

                    temp->setNext(before);

                }

            }

            newTail->setNext(current);

        }

};

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 indexes MxN to reverse: ";

    int m,n;

    cin>>m>>n;

    cout<<endl<<"After reversing from M to N:"<<endl;

    flex.reverseMN(m,n);

    flex.display();

    return 0;

}

A screen shot of a computer

Description automatically generated

A black screen with white numbers

Description automatically generated

A black screen with white text

Description automatically generated

Q2:

//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;

        int size;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

            size = 0;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

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

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

            size++;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

            size++;

        }

        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);

            size++;

        }

        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;

            size--;

        }

        void removeDup(){

        if(head==nullptr)

            return;

        node\* current = head;

        while (current && current->getNext()) {

            if (current->getData() == current->getNext()->getData()) {

                node\* temp = current->getNext();

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

                delete temp;

                size--;

            }

            else

                current = current->getNext();

        }

        tail = current;

    }

};

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<<endl<<"After removing duplicates:"<<endl;

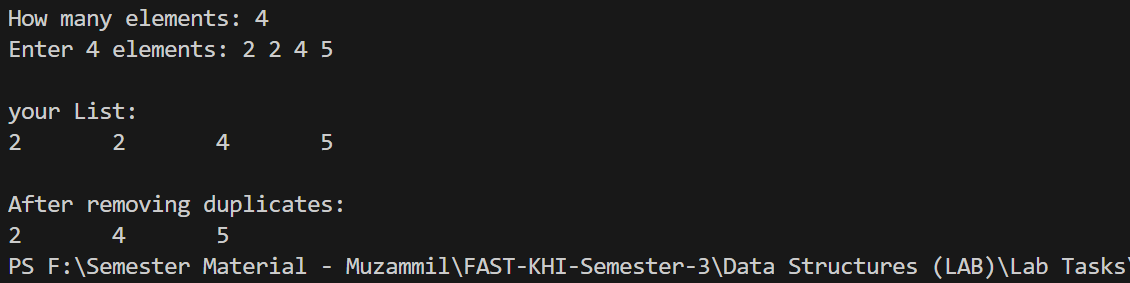
    flex.removeDup();

    flex.display();

    return 0;

}

A black screen with white text

Description automatically generated

Q3:

//23K2001 - Muzammil

#include <iostream>

using namespace std;

void swap(int &a, int &b){

    int temp = a;

    a = b;

    b = temp;

}

void sort(int \*arr1,int \*arr2,int s1,int s2){

    int index = 0;

    for(int i=0;i<s2;i++){

        int val = arr2[i];

        for (int j=0;j<s1;j++){

            if(val == arr1[j]){

                swap(arr1[index], arr1[j]);

                index++;

            }

        }

    }

}

int main() {

    int n,m;

    cout<<"Enter length of Array#1: ";

    cin>>n;

    int arr1[n];

    cout<<"Input "<<n<<" elements:"<<endl;

    for(int i=0;i<n;i++)

        cin>>arr1[i];

    cout<<"Enter length of Array#2: ";

    cin>>m;

    int arr2[m];

    cout<<"Input "<<m<<" elements:"<<endl;

    for(int i=0;i<m;i++)

        cin>>arr2[i];

    cout<<"Array#1:"<<endl;

    for(int i:arr1)

        cout<<i<<"\t";

    cout<<endl<<"Array#2:"<<endl;

    for(int i:arr2)

        cout<<i<<"\t";

    cout<<endl<<endl<<"Array#1 after sorting to Array#2:"<<endl;

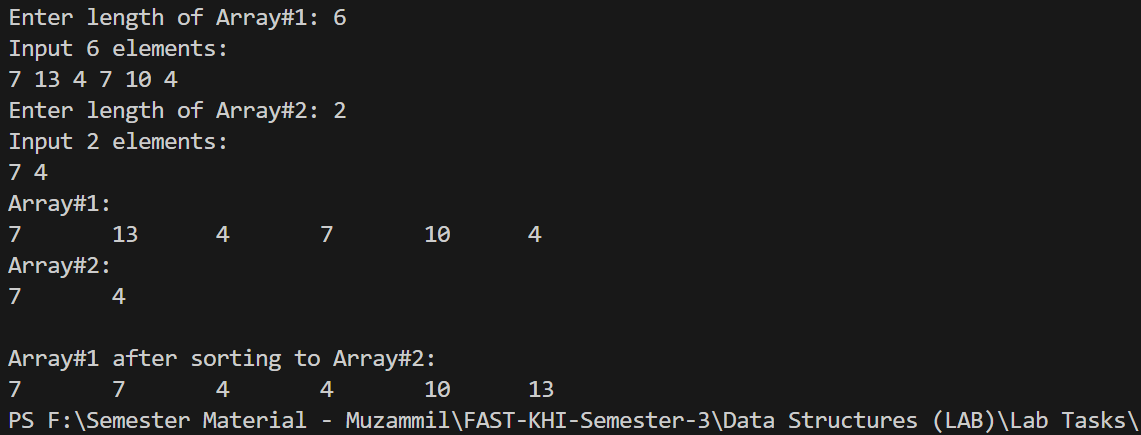
    sort(arr1,arr2,n,m);

    for(int i:arr1)

        cout<<i<<"\t";

    return 0;

}



Q4:

//23K2001 - Muzammil

#include <iostream>

using namespace std;

class stacks{

    private:

        int top,size;

    public:

        int \*arr;

        stacks():top(-1),size(0),arr(nullptr){}

        stacks(int s):top(-1),size(s){

            arr = new int[size];

            for(int i=0;i<size;i++)

                arr[i]=0;

        }

        int getSize(){ return size; }

        void push(int e){

            if(top>=(size-1)){

                cout<<"Stack overflow occured!"<<endl;

                return;

            }

            arr[++top] = e;

        }

        int pop(){

            if(top<0){

                cout<<"Stacks underflow occured!"<<endl;

                return -1;

            }

            int last = arr[top--];

            return last;

        }

        int peek(){

            if(top < 0){

                cout<<"Stack is Empty";

                return 0;

            } else{

                int x = arr[top];

                return x;

            }

        }

        void display(){

            if(isEmpty()){

                cout<<"Stack is empty!"<<endl;

                return;

            }

            for(int i=0;i<=top;i++)

                cout<<arr[i]<<" ";

            cout<<endl;

        }

        bool isEmpty(){ return (top<0); }

        ~stacks(){ delete[] arr; }

};

class queue{

    int \*arr;

    int size;

    int rear;

    int front;

public:

    queue():front(-1),rear(-1),size(0),arr(nullptr){}

        queue(int s):front(-1),rear(-1),size(s){

            arr = new int[size];

            for(int i=0;i<size;i++)

                arr[i]=-1;

        }

        void enqueue(int q){

            if(isFull()){

                cout<<"Queue is full!"<<endl;

                return;

            }

            else if(isEmpty())

                front = rear = 0;

            else

                rear++;

            arr[rear]=q;

        }

        void dequeue(){

            if(isEmpty()){

                cout<<"Queue is empty!"<<endl;

                return;

            }

            else if(front == rear)

                front = rear = -1;

            else

                front++;

        }

        bool isEmpty(){

            if(front==-1 || front>rear)

                return true;

            return false;

        }

        bool isFull(){

            if(rear==size-1)

                return true;

            return false;

        }

        int peek(){ return arr[front]; }

        ~queue(){ delete[] arr; }

};

void reverseByQueue(stacks &s){

    queue q(s.getSize());

    while (!s.isEmpty()){

        q.enqueue(s.peek());

        s.pop();

    }

    while (!q.isEmpty()){

        s.push(q.peek());

        q.dequeue();

    }

}

int main()

{

    int s;

    cout<<"Enter size of stack: ";

    cin>>s;

    stacks flex(s);

    int e;

    cout<<"Enter "<<s<<" elements:"<<endl;

    for(int i=0;i<s;i++){

        cin>>e;

        flex.push(e);

    }

    cout<<"Elements in stack: "<<endl;

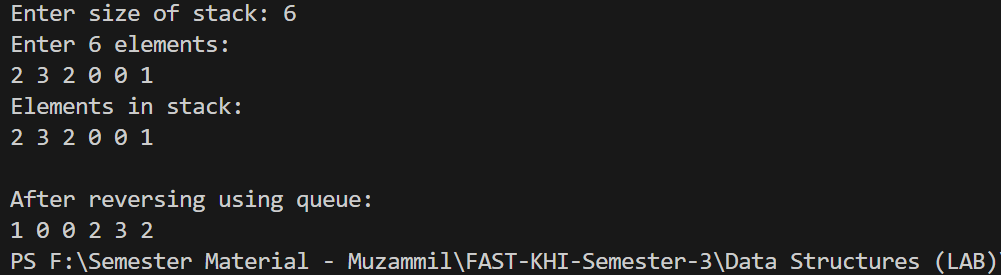
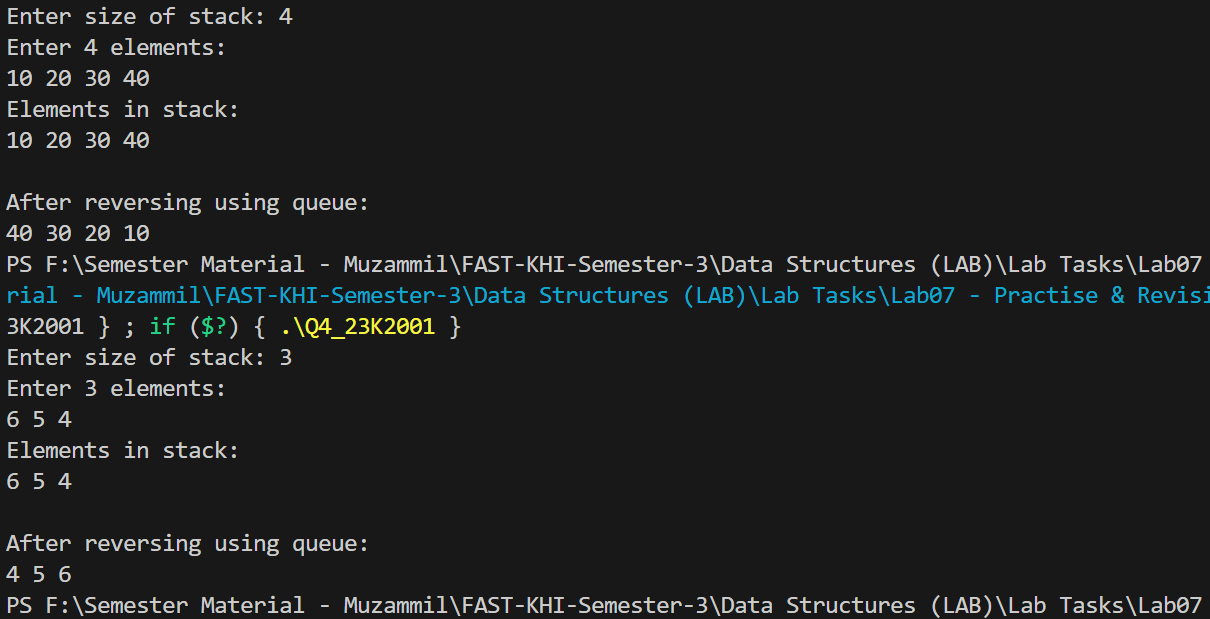
    flex.display();

    reverseByQueue(flex);

    cout<<endl<<"After reversing using queue:"<<endl;

    flex.display();

}



Q5:

//23K2001 - Muzammil

#include <iostream>

using namespace std;

class stacks{

    private:

        int top,size;

    public:

        int \*arr;

        stacks():top(-1),size(0),arr(nullptr){}

        stacks(int s):top(-1),size(s){

            arr = new int[size];

            for(int i=0;i<size;i++)

                arr[i]=0;

        }

        int getSize(){ return size; }

        void push(int e){

            if(top>=(size-1)){

                cout<<"Stack overflow occured!"<<endl;

                return;

            }

            arr[++top] = e;

        }

        int pop(){

            if(top<0){

                cout<<"Stacks underflow occured!"<<endl;

                return -1;

            }

            int last = arr[top--];

            return last;

        }

        int peek(){

            if(top < 0){

                cout<<"Stack is Empty";

                return 0;

            } else{

                int x = arr[top];

                return x;

            }

        }

        void display(){

            if(isEmpty()){

                cout<<"Stack is empty!"<<endl;

                return;

            }

            for(int i=0;i<=top;i++)

                cout<<arr[i]<<" ";

            cout<<endl;

        }

        bool isEmpty(){ return (top<0); }

        void deleteMiddle(int count=0){

            if (count == (size)/2){

                cout<<endl<<"Middle element: "<<pop()<<endl;

                return;

            }

            int val = pop();

            deleteMiddle(count + 1);

            push(val);

        }

        ~stacks(){ delete[] arr; }

};

int main(){

    int s;

    cout<<"Enter size of stack: ";

    cin>>s;

    stacks flex(s);

    int e;

    cout<<"Enter "<<s<<" elements:"<<endl;

    for(int i=0;i<s;i++){

        cin>>e;

        flex.push(e);

    }

    cout<<"Elements in stack: "<<endl;

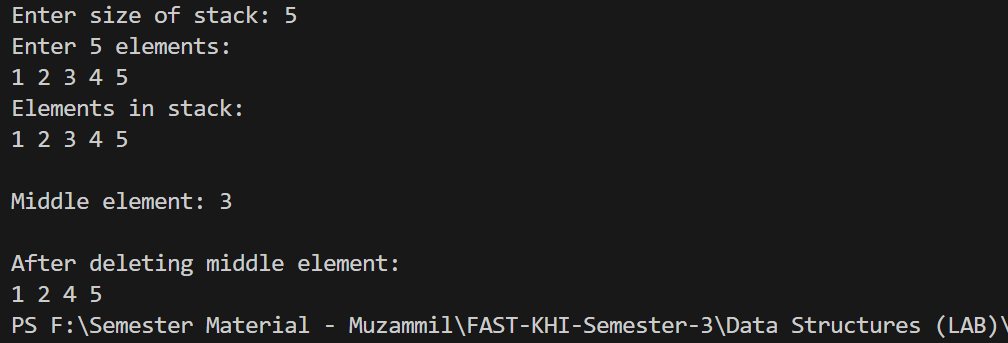
    flex.display();

    flex.deleteMiddle();

    cout<<endl<<"After deleting middle element:"<<endl;

    flex.display();

}

A computer screen shot of a black screen

Description automatically generatedA computer screen with white text

Description automatically generated

Q6:

//23K2001 - Muzammil

#include<iostream>

using namespace std;

class stacks{

    private:

        int top,size;

    public:

        char \*arr;

        stacks():top(-1),size(0),arr(nullptr){}

        stacks(int s):top(-1),size(s){

            arr = new char[size];

            for(int i=0;i<size;i++)

                arr[i]='!';

        }

        int getpeek(){ return top; }

        int getSize() { return size; }

        void push(char e){

            if(top>=(size-1)){

                cout<<"Stack overflow occured!"<<endl;

                return;

            }

            arr[++top] = e;

        }

        char pop(){

            if(top<0){

                cout<<"Stacks underflow occured!"<<endl;

                return '!';

            }

            char last = arr[top--];

            return last;

        }

        char peek(){

            if(top < 0){

                cout << "Stack is Empty";

                return 0;

            } else{

                int x = arr[top];

                return x;

            }

        }

        bool isEmpty(){ return (top<0); }

};

int precedence(char c){

    if(c == '^')

        return 3;

    else if(c == '\*' || c == '/')

        return 2;

    else if(c == '+' || c == '-')

        return 1;

    else

        return -1;

}

string infixToPostfix(string infix){

    string postfix = "";

    stacks s(infix.length());

    for(int i = 0; i < infix.length(); i++){

        char c = infix[i];

        if((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))

            postfix += c;

        else if(c == '(')

            s.push(c);

        else if(c == ')'){

            while(!s.isEmpty() && s.peek() != '('){

                char op = s.pop();

                postfix += op;

            }

            if(s.peek() == '(')

                s.pop();

        } else{

            while(!s.isEmpty() && precedence(c) <= precedence(s.peek())){

                char op = s.pop();

                postfix += op;

            }

            s.push(c);

        }

    }

    while(!s.isEmpty()){

        char op = s.pop();

        postfix += op;

    }

    return postfix;

}

int main(){

    cout<<"\t\tWelcome to Infix to Postfix expression converter!"<<endl;

    cout<<endl<<"Enter you infix expression: "<<endl;

    string e;

    cin>>e;

    cout<<endl<<"After conversion :-"<<endl;

    cout<<"Postfix: "<<infixToPostfix(e)<<endl;

    // “((A + B) – C \* (D / E)) + F”

    return 0;

}

A black screen with white text

Description automatically generatedA black screen with white text

Description automatically generated

Q7:

//23K2001 - Muzammil

#include <iostream>

using namespace std;

void print(int i,int upto){

    if(i<=upto){

        if(i%2==0)

            cout<<i-1;

        else

            cout<<i+1;

    }

    else

        return;

    i++;

    cout<<"\t";

    print(i,upto);

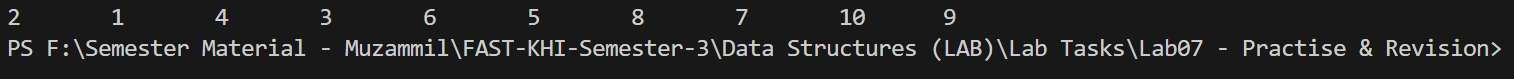
}

int main(){

    print(1,10);

    return 0;

}



Q8:

//23K2001 - Muzammil

#include <iostream>

using namespace std;

int seqA(int upto){

    if(upto==1)

        return 1;

    return upto+seqA(upto-1);

}

int seqB(int x){

    if(x==0 || x==1)

        return x;

    return seqB(x-1)+seqB(x-2);

}

int main() {

    cout<<"Sequence A:"<<endl;

    seqA(10);

    for(int i=1;i<=10;i++)

        cout<<seqA(i)<<"\t";

    cout<<endl<<endl;

    cout<<"Sequence B:"<<endl;

    for(int i=0;i<10;i++)

        cout<<seqB(i)<<"\t";

    return 0;

}

A black background with white text

Description automatically generated

Q9:

//23K2001 - Muzammil

#include <iostream>

using namespace std;

bool checkPrime(int x,int i=2){

    if(x<2)

        return false;

    if(i\*i>x)

        return true;

    if(x%i==0)

        return false;

    return checkPrime(x,i+1);

}

void composite\_primes(int m, int n) {

    if(m>n) return;

    if(checkPrime(m))

        cout<<m<<"(P)"<<endl;

    else if(m>1)

        cout<<m<<"(C)"<<endl;

    else

        cout<<m<<"(Neither)"<<endl;

    composite\_primes(m + 1, n);

}

int main() {

    composite\_primes(0,15);

    return 0;

A black screen with white text

Description automatically generated}