Question 1

#include <iostream>

using namespace std;

class node

{

public:

int data;

node \*prev;

node \*next;

node(int d)

{

this->data = d;

this->next = NULL;

this->prev = NULL;

}

void create(node \*&head, int d)

{

node \*n = new node(d);

if (head == NULL)

{

head = n;

return;

}

node \*temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = n;

n->prev = temp;

}

void print(node \*&head)

{

node \*temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

cout << "NULL\n";

}

void del(node \*&head)

{

int d;

cout << "enter the data to delete : ";

cin >> d;

if (head == NULL)

{

cout << "empty list ";

return;

}

if (head->data == d)

{

node \*to = head;

head = head->next;

delete (to);

head->prev = NULL;

return;

}

node \*temp = head;

while (temp->next != NULL && temp->next->data != d)

{

temp = temp->next;

}

if (temp->next == NULL)

{

cout << "data is not found in list \n";

return;

}

node \*to = temp->next;

temp->next = to->next;

if (to->next != NULL)

to->next->prev = temp;

delete (to);

}

};

int main()

{

int n;

cout << "enter the size of doubly link list : ";

cin >> n;

node \*head = NULL;

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

{

int d;

cout << "enter the data ";

cin >> d;

head->create(head, d);

}

head->print(head);

while (n)

{

head->del(head);

cout << "continue to 1 else 0 ";

cin >> n;

head->print(head);

}

}

Question 2

#include <bits/stdc++.h>

using namespace std;

class node

{

public:

int data;

node \*prev;

node \*next;

static int mi;

node(int d)

{

this->data = d;

this->next = NULL;

this->prev = NULL;

}

void create(node \*&head, int d)

{

node \*n = new node(d);

if (head == NULL)

{

head = n;

mi = d;

return;

}

node \*temp = head;

while (temp->next != NULL)

temp = temp->next;

mi = max(mi, d);

temp->next = n;

n->prev = temp;

}

int max\_element(node \*&head)

{

if (head == NULL)

{

return -1;

}

return mi;

}

void print(node \*&head)

{

node \*temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

cout << "NULL\n";

}

};

int node::mi = INT\_MIN;

int main()

{

int n;

cout << "enter the size of doubly link list : ";

cin >> n;

node \*head = NULL;

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

{

int d;

cout << "enter the data ";

cin >> d;

head->create(head, d);

}

head->print(head);

cout << "max element : ";

cout << head->max\_element(head);

}

Question 3

#include <iostream>

using namespace std;

class mstack

{

public:

int data;

mstack \*next;

mstack(int d)

{

this->data = d;

this->next = NULL;

}

void push(mstack \*&top)

{

int d;

cout << "enter the data : ";

cin >> d;

mstack \*n = new mstack(d);

if (top == NULL)

{

top = n;

return;

}

n->next = top;

top = n;

}

int pop(mstack \*&top)

{

if (top == NULL)

{

return -1;

}

int d = top->data;

mstack \*to = top;

top = top->next;

delete (to);

return d;

}

void print(mstack \*&top)

{

mstack \*temp = top;

while (temp != NULL)

{

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

temp = temp->next;

}

}

};

class queue

{

public:

int data;

queue \*next;

queue(int d)

{

this->data = d;

this->next = NULL;

}

void enqueue(queue \*&f, queue \*&r)

{

int d;

cout << "enter the data ";

cin >> d;

queue \*n = new queue(d);

if (f == NULL)

{

f = r = n;

return;

}

r->next = n;

r = n;

}

void dequeue(queue \*&f, queue \*&r)

{

if (f == NULL)

{

cout << "underflow queue \n";

return;

}

queue \*to = f;

f = f->next;

delete (to);

}

void print(queue \*&f)

{

queue \*temp = f;

while (temp != NULL)

{

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

temp = temp->next;

}

}

int s(queue \*&f)

{

int c = 0;

queue \*temp = f;

while (temp != NULL)

{

c++;

temp = temp->next;

}

return c;

}

};

int main()

{

queue \*f = NULL;

queue \*r = NULL;

mstack \*s = NULL;

int t = 1;

while (t)

{

cout << "enter 1 to push in stack\nenter 2 to pop in stack\nenter 3 to print stack\nenter 4 to enqueue \nenter 5 to dequeue\nenter 6 to print queue\n enter 0 to exit\nenter your choise: ";

cin >> t;

switch (t)

{

case 0:

break;

case 1:

s->push(s);

break;

case 2:

cout << "pop item : " << s->pop(s) << "\n";

break;

case 3:

cout << "STACK : top->";

s->print(s);

cout << "\n";

break;

case 4:

f->enqueue(f, r);

break;

case 5:

f->dequeue(f, r);

break;

case 6:

cout << "STACK : front->";

f->print(f);

cout << "<-rear\n";

}

}

}

Question 4

#include<string>

#include <iostream>

using namespace std;

template <typename t>

class stack

{

public:

    t data;

    static int min;

    stack \*next;

    stack(t d)

    {

        this->data = d;

        this->next = NULL;

    }

    void print(stack<t> \*&top)

    {

        cout<<"stack :\n";

        stack <t> \* temp=top;

        cout<<"top-> ";

        if(top==NULL)return;

        while(temp!=NULL)

        {

            cout<<temp->data<<endl;

            temp=temp->next;

        }

    }

    void push(stack<t> \*&top, t d)

    {

        stack<t> \*n = new stack<t>(d);

        if (top == NULL)

        {

            min=d;

            top = n;

            return;

        }

        if(d<min)

        {

            // data = 2\* current mini(d)- previous mini  (mn)

            n->data=((2\*n->data)-min);

            //updata min;

            min=d;

        }

        n->next = top;

        top = n;

    }

    t tos(stack<t> \*&top)

    {

        if (top == NULL)

            return -1;

        else

            return (top->data<min)?min:top->data; //important

    }

    t pop(stack<t> \*&top)

    {

        if (top == NULL)

            return -1;

        int a = top->data;

        stack<t> \*to\_delete = top;

        top = top->next;

        if(a<min)

        {

            int y=min;

            min = 2\*min - a;

            a=min;

        }

        delete (to\_delete);

        return a;

    }

    int minimun\_element(stack\*& s)

    {

        return min;

    }

    bool isempty(stack \*&top)

    {

        if(top == NULL) return 1;

        else return 0;

    }

};

template <typename t>

int stack<t>::min=10000007;

int main()

{

    cout<<rand()/10000;

    stack<int> \* s=NULL;

    s->push(s,3);

    s->push(s,5);

    s->push(s,2);

    s->push(s,1);

    s->push(s,1);

    s->push(s,-1);

    s->print(s);

    while(!s->isempty(s))

    {

        cout<<s->tos(s)<<endl;

        cout<<" minimun element "<<s->minimun\_element(s)<<" \n";

        s->pop(s);

    }

}

Question 5

#include<iostream>

using namespace std;

int arr[100]={0};

void inorder(int pos)

{

int i=(2\*pos)+1,j=(2\*pos)+2;

if(arr[i]!=0)inorder(i);

cout<<arr[pos]<<" ";

if(arr[j]!=0)inorder(j);

}

void preorder(int pos)

{

int i=2\*pos+1,j=2\*pos+2;

cout<<arr[pos]<<" ";

if(arr[i]!=0)preorder(i);

if(arr[j]!=0)preorder(j);

}

void postorder(int pos)

{

int i=2\*pos+1,j=2\*pos+2;

if(arr[i]!=0)postorder(i);

if(arr[j]!=0)postorder(j);

cout<<arr[pos]<<" ";

}

int main()

{

int count=0;

int d,i;

int c=1;

while(c)

{

cout<<"enter 1 to insert element\nenter 2 to delete\nenter 3 to search\nenter 4 for preorder traversal\nenter 5 for inorder traversal\nenter 6 for postorder traversal\nenter your choice ";

cin>>c;

switch(c)

{

case 1:

cout<<"\nenter data ";

cin>>d;

arr[count++]=d;

break;

case 2:

cout<<"\nenter the element to delete ";

cin>>d;

for( i=0;i<count;i++)

{

if(arr[i]==d)

{

arr[i]=0;

break;

}

}

if(i==count)cout<<"\nthe element is not found \n";

break;

case 3:

cout<<"\nenter the element to search ";

cin>>d;

for( i=0;i<count;i++)

{

if(arr[i]==d)

{

cout<<"\npresent : \n";

break;

}

}

if(i==count)cout<<"the element is not found \n ";

break;

case 4:

cout<<"preorder : ";

preorder(0);

cout<<"\n";

break;

case 5:

cout<<"inorder : ";

inorder(0);

cout<<"\n";

break;

case 6:

cout<<"postorder : ";

postorder(0);

cout<<"\n";

break;

}

}

}

Question 6

inorder :3 2 5 4 1 7 6 9 10 8 11

preorder :1 2 3 4 5 6 7 8 9 10 11

postorder:3 5 4 2 7 10 9 11 8 6 1