Question 1

#include <iostream>

#include <stack>

using namespace std;

class queue

{

public:

int data;

queue \*next;

queue(int d)

{

this->data = d;

this->next = NULL;

}

void enqueue(queue \*&f, queue \*&r, int 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";

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;

}

cout << endl;

}

int s(queue \*&f)

{

int c = 0;

queue \*temp = f;

while (temp != NULL)

{

c++;

temp = temp->next;

}

return c;

}

int s(queue \*&f)

{

queue \*i = f;

int c = 0;

while (i != NULL)

{

c++;

i = i->next;

}

return c;

}

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

{

int fs = f->data;

int ls = r->data;

stack<int> a;

stack<int> b;

int c = 1;

int n = f->s(f);

queue \*temp = f;

f->dequeue(f, r);

if (f->s(f) % 2 != 0)

{

while (c < ((n) / 2) - 1)

{

a.push(f->data);

f->dequeue(f, r);

c++;

}

}

else

{

while (c < ((n) / 2))

{

a.push(f->data);

f->dequeue(f, r);

c++;

}

}

while (c < n - 1)

{

b.push(f->data);

f->dequeue(f, r);

c++;

}

f->dequeue(f, r);

f->enqueue(f, r, fs);

while (!a.empty() || !b.empty())

{

if (!b.empty())

{

f->enqueue(f, r, b.top());

b.pop();

}

if (!a.empty())

{

f->enqueue(f, r, a.top());

a.pop();

}

}

f->enqueue(f, r, ls);

}

};

int main()

{

queue \*f = NULL;

queue \*r = NULL;

int n;

cout << "enter the length link list : ";

cin >> n;

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

{

int c;

cout << "enter the data : ";

cin >> c;

f->enqueue(f, r, c);

}

f->print(f);

f->solve(f, r);

f->print(f);

}

Question 2

#include <iostream>

#include<fstream>

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)

{

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;

}

}

};

int main()

{

ifstream m;

m.open("input.txt");

mstack \*s = NULL;

int n=1;

while(n>=0)

{

m>>n;

if(n>=0)

s->push(s,n);

}

mstack\* temp=s;

int c=0;

while(temp!=NULL)

{

c++;

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

temp=temp->next;

}

if(c!=5)

{

cout<<"error\n";

}

cout<<"\n";

while(!m.eof())

{

m>>n;

s->push(s,n);

}

s->print(s);

}

Question 3

#include <iostream>

#include <stack>

using namespace std;

class queue

{

public:

string data;

queue \*next;

queue(string d)

{

this->data = d;

this->next = NULL;

}

void enqueue(queue \*&f, queue \*&r, string 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";

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;

}

cout << endl;

}

int s(queue \*&f)

{

int c = 0;

queue \*temp = f;

while (temp != NULL)

{

c++;

temp = temp->next;

}

return c;

}

void bubbleSort(queue \*&f)

{

queue \*i = f;

while (i != NULL)

{

queue \*j = i;

string temp = i->data;

queue \*miin = i;

while (j != NULL)

{

if (j->data < miin->data)

miin = j;

j = j->next;

}

i->data = miin->data;

miin->data = temp;

i = i->next;

}

}

queue \*rev(queue \*&f)

{

if (f == NULL || f->next == NULL)

{

return f;

}

queue \*ch = rev(f->next);

f->next->next = f;

f->next = NULL;

return ch;

}

};

int main()

{

queue \*f = NULL;

queue \*r = NULL;

int n;

cin >> n;

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

{

string ss;

cout << "enter data : ";

cin >> ss;

f->enqueue(f, r, ss);

}

f->print(f);

f->bubbleSort(f);

f->print(f);

f = f->rev(f);

f->print(f);

}

Question 4

#include <iostream>

#include <stack>

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)

{

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;

}

}

void solve(mstack \*&top)

{

int min = 1000;

mstack \*top2 = NULL;

int d = top->pop(top);

if (d < min)

{

if (min != 1000)

{

top2->push(top2, min);

min = d;

}

else

{

min = d;

}

}

else

{

top2->push(top2, d);

}

d = top->pop(top);

while (d != -1)

{

if (d < min)

{

if (min != 1000)

{

top2->push(top2, min);

min = d;

}

else

{

min = d;

}

}

else

{

top2->push(top2, d);

}

d = top->pop(top);

}

//top2->print(top2);

d=top2->pop(top2);

while(d!=-1)

{

top->push(top,d);

d=top2->pop(top2);

}

top->push(top,min);

}

};

class queue

{

public:

int data;

queue \*next;

queue(int d)

{

this->data = d;

this->next = NULL;

}

void enqueue(queue \*&f, queue \*&r, int 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";

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;

}

cout << endl;

}

int s(queue \*&f)

{

int c = 0;

queue \*temp = f;

while (temp != NULL)

{

c++;

temp = temp->next;

}

return c;

}

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

{

int min=100;

queue \* rr=r;

while(f!=rr)

{

int d=f->data;

f->dequeue(f,r);

if(d<min)

{

if(min==100)

{

min=d;

}

else

{

f->enqueue(f,r,min);

min=d;

}

}

else

{

f->enqueue(f,r,d);

}

}

f->enqueue(f,r,min);

rr=r;

while(f!=rr)

{

f->enqueue(f,r,f->data);

f->dequeue(f,r);

}

}

};

int main()

{

queue \*f = NULL;

queue \*r = NULL;

int n;

cout << "enter the length queue: ";

cin >> n;

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

{

int c;

cout << "enter the data : ";

cin >> c;

f->enqueue(f, r, c);

}

f->print(f);

f->solve(f,r);

f->print(f);

mstack \*s = NULL;

int n1;

cout << "enter the length stack ";

cin >> n1;

for (int i = 0; i < n1; i++)

{

int c;

cout << "enter the data : ";

cin >> c;

s->push(s,c);

}

cout<<"initial stack : ";

s->print(s);

s->solve(s);

cout<<"\nfinal stack : ";

s->print(s);

}