//Simple using one extra stack

#include<iostream>

#include<stack>

using namespace std;

void transfer(stack<int> &s1,stack<int> &s2,int n){

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

s2.push(s1.top());

s1.pop();

}

}

void reverseStack(stack<int> &s1){

stack<int> s2;

int n=s1.size();

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

int x=s1.top();

s1.pop();

transfer(s1,s2,n-i-1);

s1.push(x);

transfer(s2,s1,n-i-1);

}

}

int main(){

stack<int> s;

for(int i=1;i<=5;i++){

s.push(i);

}//stack will contain element like 1,2,3,4,5 where 1 is at he bottom and 5 is at top

//so when we print stack element 5 will come first then 4....1

//so we need to reverse the stack elements,so that output whould be 1,2,3,4,5

reverseStack(s);

while(!s.empty()){

cout<<s.top()<<" ";

s.pop();

}

}

//Using recursion

#include<iostream>

#include<stack>

using namespace std;

void insertAtBottom(stack<int> &s,int x){

if(s.empty()){

s.push(x);

return;

}

int y=s.top();

s.pop();

insertAtBottom(s,x);

s.push(y);

}

void reverseStackRec(stack<int> &s){

if(s.empty())

return;

int x=s.top();

s.pop();

reverseStackRec(s);

insertAtBottom(s,x);

}

int main(){

stack<int> s;

for(int i=1;i<=5;i++){

s.push(i);

}

reverseStackRec(s);

while(!s.empty()){

cout<<s.top()<<" ";

s.pop();

}

}