# include <iostream>

#include<string>

using namespace std;

class Node {

public:

int num;

Node\* next;

};

class Stack {

public:

Node\* top = NULL;

void push(char n) { // insert

Node\* new\_node = new Node();

new\_node->num = n;

// (top)node -> 6 -> 3

new\_node->next = top;

top = new\_node;

}

char pop() {

if (top != NULL) {

char value = top->num;

top = top->next;

return value;

}

else {

return 1;

}

}

char to()

{

if (top != NULL) {

char value = top->num;

return value;

}

else {

return '\0';

}

}

bool is\_empty() {

if (top == NULL) {

return true;

}

else {

return false;

}

}

void print\_stack() {

Node\* current = top;

while (current != NULL) {

cout << current->num << " - ";

current = current->next;

}

cout << endl;

}

};

bool isbalanced(string s){

Stack st;

string m=s;

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

int position =0;

char c=m[i];

if(c=='['||c=='('||c=='{'){

st.push(c);

}

else if(c==']'||c==')'||c=='}')

{

if(st.is\_empty()){

cout<<"unbalanced at "<<position;

return false;

}

bool em= st.is\_empty();

if(em!='\0'){

char pop= st.pop();

if(!((c=='['&&pop==']')||(c=='{'&&pop=='}')||(c=='('&&pop==')'))){

cout<<"The expression is unbalanced at position "<<position;

return false;

}

}

}

position++;

}

if (!st.is\_empty()) {

cout << "The expression is unbalanced ";

while (!st.is\_empty()) {

cout << st.pop() << " ";

}

cout << endl;

return false;

}

return true;

}

int main() {

string str="({)";

if(isbalanced(str)){

cout<<"Balanced ";

}

else{

cout<<"unbalanced ";

}

return 0;

}

