Ex.no:05 Stack implementation & Two-way stack implementation

Date:30.07.24

Aim:

Program:

1.)Given a continuous string of containing only variables,open and closed parenthesis,find whether the parenthesis are balanced.Use stack for implementing the logic.

2.) Given a string s, remove duplicate letters so that every letter appears once and only once. You must make sure your result is the smallest in lexicographical order among all possible results

3.) You are given a string s, which contains stars *.In one operation, you can choose a star in s. Remove the closest non-star character to its left, as well as remove the star itself.Return the string after all stars have been removed.

Algorithm:

Code:

1.)

```
#include<string>
#include<algorithm>
#include<iostream>
using namespace std;
class life{
    int top;
    public:
    char st[100];
    life(){
        top=-1;}
    bool push(char a){
        if(top>=100-1){
            cout<<"Stack overflow";
            return false;}
        else{
            st[++top]=a;
            return true;}}
    bool isempty(){
        return (top<0);}
    char pop(){
      if(top<0){
            cout<<"LOP";
        return 'f';}
      else{
          char a=st[top--];
        return a;}}
};
string problem(string s1)
```

```
{
    life st;
    for(char c:s1){
        if(c=='{'){
            st.push(c);}
        else if(c=='}'){
                if (st.isempty()) {
                return "Unbalanced";}
            char ch=st.pop();
            if(ch!='{'){
                return "Unbalanced";}}}
    return st.isempty() ? "Balanced" : "Unbalanced";
}
int main(){
    life s1;
    string s;
    cout<<"Enter the string:";
    cin>>s;
    cout<<problem(s);
}
```

Output:

```
Enter the string:{a+b}+{c
Unbalanced
Process returned 0 (0x0)   execution time : 28.763 s
Press any key to continue.
```

2.)

```
#include <iostream>
using namespace std;
string removeDuplicateLetters(string s){
 int cnt[26]=0;vis[26]=0;
int n = s.size();
for (int i = 0; i < n; i++)
        cnt[s[i] - 'a']++;
string res = " ";
for (int i = 0; i < n; i++) {
    cnt[s[i] - 'a']--;
        if (!vis[s[i] - 'a']) {
while (res.size() > 0 && res.back() > s[i] && cnt[res.back() - 'a'] > 0){
        vis[res.back() - 'a'] = 0;
        res.pop_back();}
res += s[i];
vis[s[i] - 'a'] = 1;}}
return res;}
int main()
{
    string S;
    cout<<"Enter a string:";
    cin>>S;
    cout << removeDuplicateLetters(S);
return 0;}
```

Output:

```
Enter a string:arunsriram
 arunsim
Process returned 0 (0x0)   execution time : 4.246 s
Press any key to continue.
```

3.)

```
#include<algorithm>
#include<iostream>
using namespace std;
class life{
    public:
    char st[1000];
    int top;
    life(){
        top=-1;}
    void push(int a){
        if(top>=1000-1){
            cout<<"Stack overflow";}
        else{
            st[++top]=a;}}
    bool isempty(){
    if(top==-1){
        return true;}
    else{
        return false;}}
    int pop(){
      if(top<0){
        return -1;}
      else{
        int a=st[top];
        top--;
        return a;}}};
void display(life s1){
    int a=s1.top;
    while(a!=-1){
```

```
        cout<<s1.st[a];
        a--;}}
life problem(life st){
    int b=0;
    int a=st.top;
    while(a!=-1){
    if(st.st[a]=='*'){
        if(st.st[a+1]!='/'){
            st.st[a+1]='/';
            st.st[a]='/';
            a--;}
        else{
                st.st[a]='/';
                b=a+1;
        while(st.st[b]=='/'){
            b++;}
        st.st[b]='/';
        a--;}}
    else{
        a--;}}
    return st;}
string constr(life s2){
    string strin="";
    int a=s2.top;
    while(a!=-1){
        if(s2.st[a]!='/'){
            strin+=s2.st[a];}
        a--;}
    return strin;
}
```

```
int main(){
    life s1;
    s1.top=-1;
    string s;
    cout<<"Enter the string:";
    cin>>s;
    reverse(s.begin(),s.end());
for(char c:s){
        s1.push(c);
}
life s2=problem(s1);
    display(s2);
    string result=constr(s2);
    cout<<endl<<result;
}
```

Output:

```
Enter the string:leet*co*de
lee//c//de
leecde
Process returned 0 (0x0)   execution time : 12.772 s
Press any key to continue.
```

Result:

The above programs are executed successfully.