***Week – 6 (24.05.2021 – 29.05.2021)***

***RANDOM CODES***

1. ***Add Binary:***

class Solution {

public:

string addBinary(string a, string b) {

int i = a.size()-1, j = b.size()-1, carry = 0, sum=0;

string res="";

while(i>=0 || j>=0)

{

sum = carry;

carry = 0;

if(i>=0)

{

sum = sum + (a[i] - 48);

i--;

}

if(j>=0)

{

sum = sum + (b[j] - 48);

j--;

}

carry = sum/2;

sum = sum%2;

res = (char)(sum + 48) + res;

}

if(carry != 0) res = (char)(carry + 48) + res;

return res;

}

};

1. ***Evaluate Reverse Polish Notation:***

class Solution {

public:

string oper(int n1, int n2, string op)

{

if(op == "+") return to\_string(n2+n1);

if(op == "-") return to\_string(n2-n1);

if(op == "\*") return to\_string(n2\*n1);

if(op == "/") return to\_string(n2/n1);

return "";

}

int evalRPN(vector<string>& tokens) {

stack<string> val;

int n1, n2, i;

string n;

for(i=0; i<tokens.size(); i++)

{

if(tokens[i]=="+" || tokens[i]=="-" || tokens[i]=="\*" || tokens[i]=="/")

{

n1 = stoi(val.top());

val.pop();

n2 = stoi(val.top());

val.pop();

n = oper(n1, n2, tokens[i]);

val.push(n);

}

else val.push(tokens[i]);

}

return stoi(val.top());

}

};

1. ***Ransom Note:***

class Solution {

public:

bool canConstruct(string ransomNote, string magazine) {

vector<int> v(26,0);

for(char c : magazine)

v[c - 'a']++;

for(char c : ransomNote)

{

if(v[c-'a']==0)

return false;

v[c-'a']--;

}

return true;

}

};

1. ***Check If Word Is Valid After Substitutions:***

class Solution {

public:

bool isValid(string s) {

int i;

stack<char> st;

for(i=0; i<s.size(); i++)

{

if(s[i] == 'a') st.push(s[i]);

else if(s[i] == 'b')

{

if(st.empty()) return false;

if(st.top() == 'a') st.pop();

else return false;

st.push(s[i]);

}

else

{

if(st.empty()) return false;

if(st.top() == 'b') st.pop();

else return false;

}

}

if(st.empty()) return true;

else return false;

}

};

1. ***Simplify Path:***

class Solution {

public:

string simplifyPath(string path) {

stack<string> st;

string s = "", res = "";

int i;

for(i=0; i<path.size(); i++)

{

if(path[i] == '/') continue;

s = "";

while(i < path.size() && path[i] != '/')

{

s = s + path[i];

i++;

}

if(s == "..")

{

if(!st.empty())

st.pop();

}

else if(s == ".") continue;

else st.push(s);

}

while(!st.empty())

{

res = "/" + st.top() + res;

st.pop();

}

if(res == "") res = "/";

return res;

}

};

1. ***Remove Outermost Parentheses:***

class Solution {

public:

string removeOuterParentheses(string s) {

stack<char> st;

int i;

string res = "";

char temp;

for(i=0; i<s.size(); i++)

{

if(s[i] == '(')

{

if(!st.empty()) res = res + "(";

st.push(s[i]);

}

if(s[i] == ')')

{

temp = st.top();

st.pop();

if(st.empty()) continue;

else res = res + ")";

}

}

return res;

}

};

1. ***Make the String Great:***

class Solution {

public:

string makeGood(string s) {

stack<char> st;

int i;

string res = "";

for(i=0; i<s.size(); i++)

{

if(!st.empty() && islower(st.top()) && toupper(st.top()) == s[i]) st.pop();

else if(!st.empty() && isupper(st.top()) && tolower(st.top()) == s[i]) st.pop();

else st.push(s[i]);

}

while(!st.empty())

{

res = st.top() + res;

st.pop();

}

return res;

}

};

1. ***Baseball Game:***

class Solution {

public:

int calPoints(vector<string>& ops) {

int i, n1, n2, sum=0;

stack<int> s;

for(i=0; i<ops.size(); i++)

{

if(ops[i] == "C") s.pop();

else if(ops[i] == "D") s.push(2 \* s.top());

else if(ops[i] == "+")

{

n1 = s.top();

s.pop();

n2 = s.top();

s.push(n1);

s.push(n1 + n2);

}

else s.push(stoi(ops[i]));

}

while(!s.empty())

{

sum = sum + s.top();

s.pop();

}

return sum;

}

};

1. Crawler Log Folder:

class Solution {

public:

int minOperations(vector<string>& logs) {

stack<string> s;

int i, t=0;

for(i=0; i<logs.size(); i++)

{

if(logs[i] == "./") continue;

else if(logs[i] == "../" && !s.empty()) s.pop();

else if(logs[i] == "../" && s.empty()) continue;

else s.push(logs[i]);

}

while(!s.empty())

{

t++;

s.pop();

}

return t;

}

};

1. ***Backspace String Compare:***

class Solution {

public:

bool backspaceCompare(string s, string t) {

stack<char> st1, st2;

int i;

string s1, s2;

for(i=0; i<s.size(); i++)

{

if(!st1.empty() && s[i] == '#') st1.pop();

else if(st1.empty() && s[i] == '#') continue;

else st1.push(s[i]);

}

for(i=0; i<t.size(); i++)

{

if(!st2.empty() && t[i] == '#') st2.pop();

else if(st2.empty() && t[i] == '#') continue;

else st2.push(t[i]);

}

while(!st1.empty() && !st2.empty())

{

if(st1.top() != st2.top()) return false;

st1.pop();

st2.pop();

}

if(st1.empty() && st2.empty()) return true;

return false;

}

};

1. ***Goat Latin:***

class Solution {

public:

string toGoatLatin(string sentence) {

stack<string> s;

int i, n=0, m;

string str="", res = "";

char t;

for(i=0; i<sentence.size(); i++)

{

if(sentence[i] == ' ')

{

s.push(str);

str = "";

n++;

}

else str = str + sentence[i];

if(i == sentence.size()-1)

{

s.push(str);

str = "";

n++;

}

}

while(!s.empty())

{

str = s.top();

m = n;

if(str[0]=='a' || str[0]=='e' || str[0]=='i' || str[0]=='o' || str[0]=='u' || str[0]=='A' || str[0]=='E' || str[0]=='I' || str[0]=='O' || str[0]=='U')

str = str + "ma";

else

{

t = str[0];

str.erase(str.begin());

str = str + t +"ma";

}

while(m>0)

{

str = str + "a";

m--;

}

res = str + " " + res;

n--;

s.pop();

}

res.erase(res.begin()+res.size()-1);

return res;

}

};

1. ***Unique Morse Code Words:***

class Solution {

public:

int uniqueMorseRepresentations(vector<string>& words) {

vector<string> morse = {".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."};

vector<string> res;

int i, j;

string s;

for(i=0; i<words.size(); i++)

{

s = "";

for(j=0; j<words[i].size(); j++)

{

s = s + morse[words[i][j]-'a'];

}

res.push\_back(s);

}

if(res.size() == 1) return res.size();

sort(res.begin(), res.end());

res.erase(unique(res.begin(), res.end()), res.end());

return res.size();

}

};

1. ***Reach a Number:***

class Solution {

public:

int reachNumber(int target) {

target = abs(target);

int i;

for (i = 1; target; i++) {

target -= i;

if(target <= 0 && ~target & 1)

return i;

}

return 0;

}

};

1. ***Robot Bounded In Circle:***

class Solution {

public:

bool isRobotBounded(string instructions) {

char direction = 'N';

int xCord = 0, yCord = 0;

for (char instruction : instructions)

{

if (instruction == 'G')

{

if (direction == 'N' || direction == 'S')

yCord += direction == 'N' ? 1 : -1;

else

xCord += direction == 'E' ? 1 : -1;

}

else

{

if (direction == 'N')

direction = instruction == 'L' ? 'W' : 'E';

else if (direction == 'W')

direction = instruction == 'L' ? 'S' : 'N';

else if (direction == 'S')

direction = instruction == 'L' ? 'E' : 'W';

else

direction = instruction == 'L' ? 'N' : 'S';

}

}

return ((xCord == 0 && yCord == 0) || direction != 'N');

}

};

1. ***Distribute Candies to People:***

class Solution {

public:

vector<int> distributeCandies(int candies, int num\_people) {

vector<int>arr(num\_people,0);

int num=1,i=0;

while(candies>0)

{

if(i==num\_people)

i=0;

else

{

if(candies>num)

{

arr[i]+=num;

candies-=num;

}

else

{ arr[i]+=candies;

candies=0;

}

num++;

i++;

}

}

return arr;

}

};