BITS F232: FOUNDATIONS OF DATA STRUCTURES & ALGORITHMS (1ST SEMESTER 2023-24)

STACK ADT CONTINUED…

Chittaranjan Hota, PhD
Sr. Professor of Computer Sc.
BITS-Pilani Hyderabad Campus
hota[AT]hyderabad.bits-pilani.ac.in

STACK USAGE EXAMPLES

Let *S* be an empty stack
for *i*=0 to *n*-1 **do**
 if *X*[*i*] is an opening grouping symbol **then**
  S.push(*X*[*i*])
 else
  if *X*[*i*] is a closing grouping symbol **then**
   if *S*.empty() **then**
    return false {nothing to match with}
   if *S*.pop() does not match the type of *X*[*i*] **then**
    return false {wrong type}
if *S*.empty() **then**
  return true {every symbol matched}

else

  return false {some symbols were never matched}

```
((a + b) * c + d - e) / (f + g) - (h + j) * k - l / (m - n):  BALANCED! (0.005 ms.)
(({}{()}):  BALANCED! (0.001 ms.)
({}){()}:  BALANCED! (0.001 ms.)
(){}(}:  NOT Balanced (0 ms.)
```

(Output)

```
138         if (token == '(' || token == '{') // this is an opening bracket
139         {
140             stack.push(token);
141         }
142         else if (token == ')') // found closing parentheses
143         {
144             if (stack.empty() || stack.top() != '(')
145             {
146                 return false; // match not found
147             }
148             stack.pop(); // match found
149         }
```

Lab-6 (Next week's lab)

STL

```
#include <stack>
using std::stack;                    // make stack accessible
stack<int> myStack;                  // a stack of integers
```

STACK USAGE: REVERSING A VECTOR EXAMPLE

```
template <typename E>
void reverse(vector<E>& V) {
    ArrayStack<E> S(V.size());
    for (int i = 0; i < V.size(); i++)
        S.push(V[i]);
    for (int i = 0; i < V.size(); i++) {
        V[i] = S.top();  S.pop();
    }
}
```

Non-recursive algo…

```
Enter size of input vector : 4

Enter input vector : 2 5 7 9

Input vector : 2 5 7 9

Reversed vector : 9 7 5 2
```

STACK ADT: LINKED LIST IMPLEMENTATION

IMPLEMENTING A STACK WITH A GENERIC LINKED LIST

```
88  int LinkedStack::size() const
89  { return n; }
90
91  bool LinkedStack::empty() const
92  { return n == 0; }
93
94
95  const Elem& LinkedStack::top() {
96      if (empty()) cout<<"Top of empty stack\n";
97      return S.front();
98  }
99
100 void LinkedStack::push(const Elem& e) {
101     ++n;
102     S.addFront(e);
103 }
104
105 void LinkedStack::pop() {          // pop the stack
106     if (empty()) cout<<"Pop from empty stack\n";
107     --n;
108     S.removeFront();
109 }
```

```
template <typename E>
void SLinkedList<E>::removeFront() {
    SNode<E>* old = head;
    head = old->next;
    delete old;
}

template <typename E>
void SLinkedList<E>::traverse(){
    SNode<E> *temp = head;
    while(temp != NULL){
        cout<<temp->elem<<" ";
        temp = temp->next;
    }
    cout<<endl;
}

typedef string Elem;
class LinkedStack {
    public:
        LinkedStack();
        int size() const;
        bool empty() const;
        const Elem& top();
        void push(const Elem& e);
        void pop();
    private:
        SLinkedList<Elem> S;
        int n;
};
```

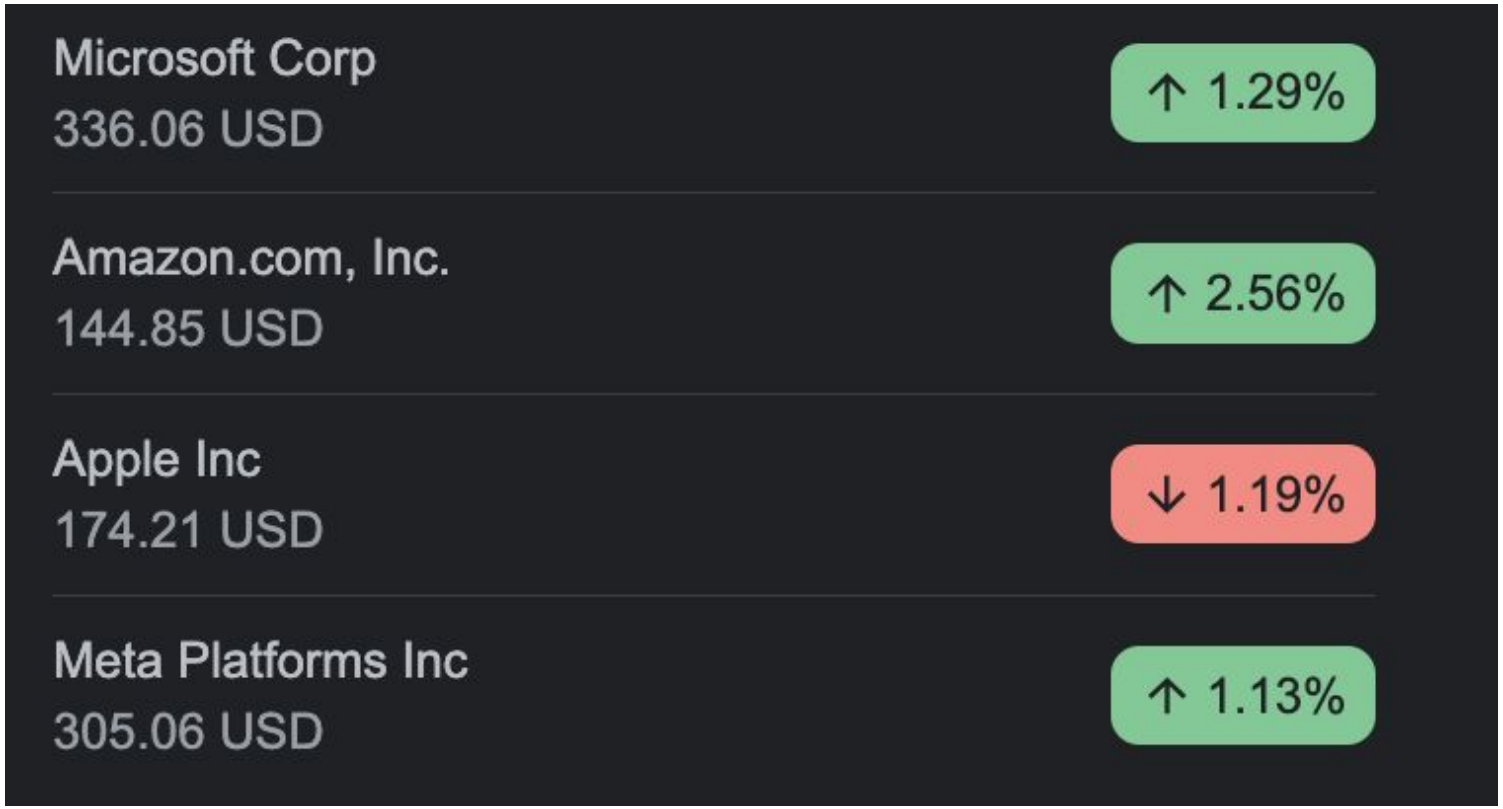
```
Enter input 1
10
Pushing : 10
Enter input 1
20
Pushing : 20
Enter input 1
30
Pushing : 30
Enter input 3
Getting top
30
Enter input 4
Getting size
3
Enter input 5
Stack is not empty
Enter input 2
Attempting pop
Enter input 3
Getting top
20
Enter input
```

STACK USAGE: MATCHING TAGS IN AN HTML DOC

```
bool isHtmlMatched(const vector<string>& tags) {
    LinkedStack S;
    typedef vector<string>::const_iterator Iter;

    for (Iter p = tags.begin(); p != tags.end(); ++p) {
        if (p->at(1) != '/')
            S.push(*p);
        else {
            if (S.empty()) return false;
            string open = S.top().substr(1);
            string close = p->substr(2);
            if (open.compare(close) != 0) return false;
            else S.pop();
        }
    }
    if (S.empty()) return true;
    else return false;
}
```

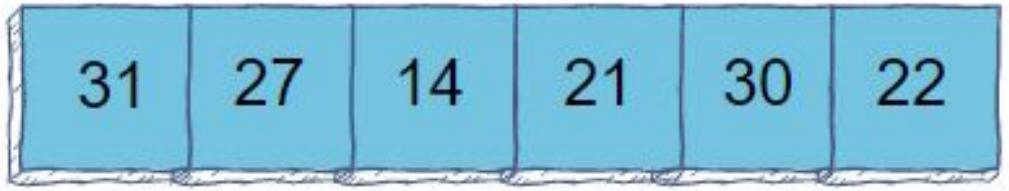
COMPUTING STOCK SPAN: STACK USAGE

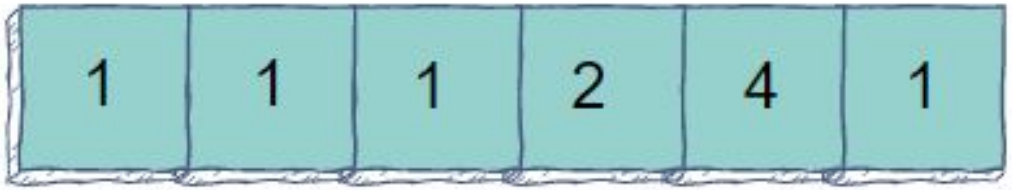


Source: Internet, 14th Sept 2023

COMPUTING STOCK SPAN CONTINUED…

X





S

```
Input Stocks Data: 6 3 4 5 2
Output Spans: 1 1 2 3 1 (0.004 ms.)
Input Stocks Data: 2 4 5 6 7 8 9
Output Spans: 1 2 3 4 5 6 7 (0 ms.)
Input Stocks Data: 100 80 60 70 60 75 85
Output Spans: 1 1 1 2 1 4 6 (0.001 ms.)
```

Algorithm *spans2*(*X*, *n*) Complexity: O(n)

A ← new array of *n* integers	*n*
S ← new empty stack	1
for *i* ← 0 **to** *n* − 1 **do**	*n*
while (¬*S.empty*() ∧ *X*[*S.top*()] ≤)	*n*
S.pop()	*n*
if *S.empty*() **then**	*n*
A[*i*] ← *i* + 1	*n*
Lab 6: Next week's Lab	
	n