Stacks

Representing stacks in C

```
#define STACKSIZE 100
struct stack {
     int top;
      int items[STACKSIZE];
};

struct stack s;
int empty(struct stack * ps)
{
   if (ps->top==-1) return (TRUE);
   else return (FALSE);
}  // end and empty
```

pop operation

```
int pop(struct stack * ps)
{
```

```
if empty (ps) {
printf("%s","Stack underflow");
exit(1);
}
return(ps->items[ps->top--]);
}

push operation
void push(struct stack *ps, int x)
{
    if (ps->top==STACKSIZE-1]{printf("%s","Stack overflow");
    exit(1);
}
else ps-> items[++(ps->top)] = x;
return;
}// end push
```

C++ implementation

C++ Implementation

```
template < class T>
class Stack{
private:
   int top;
   T * nodes;
public:
    Stack();
    bool empty(void);
    void push(T &);
    T pop(void);
    ~Stack ();    //destructor
};
```

Implementation of templates

```
template < class T > Stack<T>:: Stack()
{
   top =-1;
   nodes= new T [STACKSIZE];
};

Template<class T> Stack<T>::~Stack(){delete nodes;};
```

Empty Push and Pop

```
template <class T> bool Stack<T> : : empty(void)
{
  return (top<0);
}

template <class T > void Stack <T> :: push(T & j)
{
   if  (top == STACKSIZE-1){
     cout<,"Stack overflow"<<endl;
     return
  }// end if
nodes[++top] = j;
}

template<class T> Stack <T> :: pop(void)
{
  T p;
  if empty(){
    cout<,"Stack underflow"<<endl;
    return p;
  }
p = nodes[top--];
return p;
}
```

Infix Prefix & Postfix Expression

4 + 5 * 6 = 54 or 34 ????

This ambiguity is there in infix expression but removed in prefix and postfix expression.

A + B A ,B :operands
+ :operator
“pre” “in” and “ post” denote the relative position of the operator w.r.t operand

+AB : prefix
A+B : infix
AB+ : postfix

One point about prefix and postfix form of an expression is that it requires no parenthesis

Infix	Prefix	Postfix
A+(B*C)	+A*BC	ABC*+
(A+B)*C	*+ABC	AB+C*

Evaluating a postfix expression
6 2 3 + - 3 8 2 / + *

Symbol	Opnd1	Opnd2	Value	Opstk
6				6
2				6,2
3				6,2,3
+	2	3	5	6,5
-	6	5	1	1
3				1,3
8				1,3,8
2				1,3,8,2
/	8	2	4	1,3,4

Symbol	Opnd1	Opnd2	Value	Opstk
+	3	4	7	1,7
*	1	7	7	7