```
//Eduardo Martinez
//CS211 Lab5, assignment 3
//Stack class - header file
#ifndef STACK_H
#define STACK_H
#include <string>
using namespace std;

typedef int el_t;// the element type, el_t, is int
const int MAX = 5;//The MAX number of elements for the
stack

class Stack
{
 private:
  el_t st[MAX]; // st is an array of el_t's
  int top; // the index to the top element
  void stackError(string msg);//utility function for error
handling
 public:
  Stack(); //Constructor
  // PURPOSE: returns true if empty and false if not empty.
  bool isEmpty();
  // PURPOSE: returns true is full and false if not full
  bool isFull();
  // HOW TO CALL: pass an element to be pushed
  // PURPOSE: if not full, enters an element at the top.
  // otherwise, calls an emergency exit routine
  void push(el_t);
  // HOW TO CALL: pops last element on stack
  // PURPOSE: if empty calls an emergency exit routine
  //if not empty, removes an element from the top
  el_t pop();
  // HOW TO CALL: pops the top element on stack
  // PURPOSE: if empty calls an emergency exit routine
  //if not empty, removes an element from the top
  el_t topElem();
  // HOW TO CALL: displays all elements in stack
  // PURPOSE: uses for loop to ouput all elements in stack
  void displayAll();
  // PURPOSE: pops top element on stack until empty to
clear the stack
  void clearIt();
};
#endif
```

```
//Eduardo Martinez
//CS211 Lab 5, Assignment 3
//Stack class - implementation file
#include "stack.h"
#include <iostream>
using namespace std;

// PURPOSE: constructor which initializes top
Stack::Stack()
{
  top = -1;
}
// PURPOSE: to check if stack is empty
// ALGORITHM: checks if top is less than 0
// if true then returns true, if false returns false
bool Stack::isEmpty()
{
   if (top < 0)
     return true;
   else
     return false;
}
// PURPOSE: to check if stack is full
// ALGORITHM: checks if top is greater than or equal to MAX
- 1
// if true then returns true, if false returns false
bool Stack::isFull()
{

  if(top >=  MAX-1)
    return true;
  else
    return false;
}
// PURPOSE: to push a passed element to the stack
// PARAMS: new element n of type el_t
// ALGORITHM: if not full, increment top and place n there.
// else stackError is called
void Stack::push(el_t elem)
{
  if (isFull())//stack is full
    stackError("stack overwflow");
  else//stack is not full
    {
      st[++top] = elem;
    }
}
// PURPOSE: to pop last element on the stack
// ALGORITHM: if not empty, return top elem on stack and
decrement top.
// else stackError is called
el_t Stack::pop()
{
  if (isEmpty())
    stackError("stack underflow");
  else
    return st[top--];
}
// PURPOSE: to pop last element on the stack
// ALGORITHM: if not empty return top element on stack.
// else stackError is called
el_t Stack::topElem()
{
  if (isEmpty())
    stackError("stack is empty");
  else
    return st[top];
}
// PURPOSE: to handle unexpected errors encountered by
other methods
// PARAMS: a string message to be displayed
// ALGORITHM: simply cout the message and exit from the
program
void Stack::stackError(string msg)
{
  cout << msg << endl;
  exit(1); //ends the program. 1 is returned to the
operating system.
  //0 = end with no errors, 1 = end with errors
}
// PURPOSE: to show all elements in stack
// ALGORITHM: goes through a for loop to output elements in
stack
void Stack::displayAll()
{
  for(int i =top; i > -1; i--)
      cout << st[i] << endl;
}
// PURPOSE: to clear the stack
// ALGORITHM: while the stack is not empty it pops and item
from the stack
void Stack::clearIt()
{
  while(!isEmpty())
    pop();
}
```

```
/*
Eduardo Martinez
CS 211 Lab 5-2, Assignment 3
postfix- the purpose of the program is to get an input
expression from the user.
every character from the expression is checked if its an
operand,operator, or invalid item.
if its an operand then it pushed onto the stack until an
operator shows up. if an error
occurs then "invalid expression" is output.
*/

#include <iostream>
#include <string>
#include<cstring>
#include "stack.h"
using namespace std;



bool isOperand(char ch);
int charToInt(char ch);
bool isOperator(char ch);
int doMath(int opr1, char opt, int opr2);



int main()
{
  Stack stk;
  char curItem;
  int i=0;
  int ans;
  bool error = false; //for any errors that may occur
  string expr;
  cout << "Enter the expression: ";
  cin >> expr;
  char * cstr= new char[expr.length()+1];//turns the string
to c-string
  strcpy (cstr,expr.c_str());//copies the string into cstr


while(cstr[i] != '\0' && !error)
   {
     curItem = cstr[i];//current item from the c-string
     if(isOperand(curItem))//is its an operand
       {
      if(!stk.isFull())
        stk.push(charToInt(curItem));//pushes to stack if
stack isnt full
      else
         error =true; //stack is full
       }
     else if(isOperator(curItem))//if its an opertator then
it does the math
```

```
       {                         //and pushed on to stack
       int op1,op2;
       if(!stk.isEmpty())// pops last item from stack if its
not empty
         op1 = stk.pop();
       else
           error = true;//stack is empty
       if(!stk.isEmpty())// pops last item from stack if its
not empty
           op2 = stk.pop();
          else
           error = true;//stack is empty
       if(!stk.isFull())// checks if stack is full
         stk.push(doMath(op1,curItem,op2));// does math of
two last popped items
       else                              // and pushes
answer onto stack
           error = true;//stack is full
       }
     else //invalid item
       error = true;
     i++;//increments to get next item from c-string
   }
 if(!stk.isEmpty())//pops last item if stack isnt empty
   ans = stk.pop();
 else
         error = true;

 if(!error && stk.isEmpty())//if there arent any errors and
stack is empty then pops answer
   cout << "the answer is: " << ans << endl;
  else
    cout << "invalid expression" << endl;// if stack isnt
empty or there was an error then
                                        // the expression
is invalid

  return 0;
}

bool isOperand(char ch)//checks if its an operand
{
  if(ch <= 57 && ch >=48)// subtracts to check is its 0-9
    return true;
  else
    return false;
}
int charToInt(char ch)// converts char to int
{
  int num= 48;//ascii 0 is 48
  num = (int)ch - num;//subtract to convert to int
  return (num);
}
bool isOperator(char ch)// checks if its an operator
{
  if(ch == '+' || ch == '-' || ch == '*' || ch == '/') //ch
is an operator
    return true;
  else //ch is not an operator
    return false;
}
int doMath(int opr2, char opt, int opr1)//does math
{
  int answer;
  if(opt == '+')
    answer = opr1 + opr2;
  else if(opt == '-')
    answer = opr1 - opr2;
  else if(opt == '*')
    answer = opr1 * opr2;
  else if (opt == '/')
    answer = opr1 / opr2;

  return answer;//returns answer
}
```

Test runs:

```
[marti540@empress cs211]$ ./a.out
Enter the expression: 12345+-+-
the answer is: 5
[marti540@empress cs211]$ ./a.out
Enter the expression: 123456+-+-
invalid expression
[marti540@empress cs211]$ ./a.out
Enter the expression: 1+-
invalid expression
[marti540@empress cs211]$ ./a.out
Enter the expression: q2+
invalid expression
[marti540@empress cs211]$ ./a.out
Enter the expression: 12+3
invalid expression
```

```
/*
Eduardo Martinez
CS211 Lab 6, Assignment 3
Fibonacci- the purpose of the program is to find the
fibonacci number of given
position and output the time to see the effiecency of non-
recursives solutions
*/
#include <iostream>
#include <time.h>
using namespace std;

int fib(int x);
int fibnoacci(int x);

int main()
{
  clock_t start, end;
  int pos;
  cout << "Enter a position: ";
  cin >> pos;
  int fibNum;
  //start timing
  start = clock();
  fibNum =  fib(pos);

  end = clock();
  cout << "Elapsed time: " << (end - start) /
double(CLOCKS_PER_SEC) * 1000 << " milliseconds" << endl;
  cout << "Fibonnaci number at position " << pos << " is "
<< fibNum << endl;

 start = clock();
 fibNum = fibnoacci(pos);
 end = clock();
 cout << "Elapsed time: " << (end - start) /
double(CLOCKS_PER_SEC) * 1000 << " mil\
liseconds" << endl;
 cout << "Fibonnaci number at position " << pos << " is "
<< fibNum << endl;

  return 0;
}
int fib(int x)
{
  if (x <= 1)
    return x;
  else
    return fib(x-2)+fib(x-1);

}

int fibnoacci(int x)
{
  int ans;
  int n1= 0;
  int n2= 1;
  for (int i=0; i<x ; i++)
    {
     if(i < 1)
     {
       ans= i;
     }
     else
     {
       ans= n1 +n2;
       n1= n2;
       n2 =ans;
     }
    }
  return ans;
}
```

```
[marti540@empress cs211]$ ./a.out
Enter a position: 0
Elapsed time: 0 milliseconds
Fibonnaci number at position 0 is 0
Elapsed time: 0 milliseconds
Fibonnaci number at position 0 is 0
[marti540@empress cs211]$ ./a.out
Enter a position: 1
Elapsed time: 0 milliseconds
Fibonnaci number at position 1 is 1
Elapsed time: 0 milliseconds
Fibonnaci number at position 1 is 0
[marti540@empress cs211]$ ./a.out
Enter a position: 10
Elapsed time: 0 milliseconds
Fibonnaci number at position 10 is 55
Elapsed time: 0 milliseconds
Fibonnaci number at position 10 is 55
[marti540@empress cs211]$ ./a.out
Enter a position: 20
Elapsed time: 0 milliseconds
Fibonnaci number at position 20 is 6765
Elapsed time: 0 milliseconds
Fibonnaci number at position 20 is 6765
[marti540@empress cs211]$ ./a.out
Enter a position: 46
Elapsed time: 33060 milliseconds
Fibonnaci number at position 46 is 1836311903
Elapsed time: 0 milliseconds
Fibonnaci number at position 46 is 1836311903
[marti540@empress cs211]$
```