#ifndef ARRAY\_H

#define ARRAY\_H

class Array

{

private:

int \*p;

int size;

public:

Array() { size = 10; p = new int[size];} //Default Constructor

Array(int sz); //Constructor

~Array(); //Destructor

//Array(const Array &a); //Copy Constructor

void put(int i, int j); //Like our setElement

int get(int i); //Like out getElement

};

#endif

-----------------------------------------------------/////---------------------------------------------------------

#ifndef DATE\_H

#define DATE\_H

#include <iostream>

using namespace std;

class Date

{

// Prints the date to screen in the format 1-1-2000

private:

int day; // 1 to 28-31, depending on month and year

int month; // 1 to 12

int year; // 1000 to 2100 A.D.

int array[3];

public:

// \*\* Constructors and Destructors

// Default Constructor: Initializes date to 1-1-2000 and weekday = 7 or

Date();

Date( int month, int day, int year) ;

void setDate(int month, int day, int year);

int getDay();

int getMonth();

int getYear();

friend ostream& operator<<(ostream& screen\_out, Date date\_obj);

friend istream& operator>>(istream& in, Date& date\_obj);

Date Date::operator+(int day);

void operator=(Date right\_side); // Performs assignment of dates

// Validates and sets the date. If date is invalid, sets to default value

// of 1-1-00, and weekday = 7.

\

Date operator++(int dummy); // Increments the date (postfix operator)

Date& operator++(); // Increments the date (prefix operator)

Date operator--(int dummy); // Decrements the date (postfix operator)

Date& operator--( ); // Decrements the date (prefix operator)

Date operator+(Date val); // Adds two dates together

Date operator-(Date val); // Subtracts two dates

int& operator[](int index);

Date& returnRef(Date& c1); // Returns Refernce

// Increments this date by the specified day\_count

void operator+=(int day\_count);

// Returns the number of days in specified month, or 0 if month is invalid

// and an error message (Note: 2001 is not a leap year)

};

#endif

-----------------------------------------------------/////---------------------------------------------------------

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

class Employee

{

private:

int id;

int salary;

static int count;

const static double pi;

public:

Employee(int salary);

~Employee();

int getId();

int getSalary();

static int getCount();

};

#endif

-----------------------------------------------------/////---------------------------------------------------------

#ifndef GENLINE\_H

#define GENLINE\_H

#include "GenPoint.h"

class GenLine

{

private:

GenPoint p1;

GenPoint p2;

string desc;

public:

GenLine();

void setLine(GenPoint p1, GenPoint p2);

void setDesc(string desc);

string getDesc();

GenPoint getPoint1();

GenPoint getPoint2();

GenPoint getMidPoint();

double getLength();

void draw();

};

#endif

-----------------------------------------------------/////---------------------------------------------------------

#ifndef GENPOINT\_H

#define GENPOINT\_H

class GenPoint

{

private:

int x;

int y;

public:

GenPoint(){ x = 0; y = 0;}

void setPoint(int x, int y);

int getX();

int getY();

};

#endif

-----------------------------------------------------/////---------------------------------------------------------

#ifndef GENPOLYGON\_H

#define GENPOLYGON\_H

#include "GenLine.h"

class GenPolygon

{

private:

int no\_lines;

GenLine\* lines;

public:

GenPolygon();

GenPolygon(int no\_lines);

~GenPolygon();

GenPolygon(const GenPolygon& poly);

int getNoLines();

void setNoLines(int no\_lines);

GenLine getLineAt(int index);

void setLineAt(int index, GenLine line);

void draw();

};

#endif;

-----------------------------------------------------/////---------------------------------------------------------

// Member functions defined in "String.cpp"

#ifndef GEN\_STRING\_H

#define GEN\_STRING\_H

class GenString

{

private:

char \*string; // pointer to null-terminated array of characters

public:

// \*\* Constructors and Destructors

// Default Constructor: Initializes string to null (0)

GenString();

GenString( char \*string\_value); // Constructor: Calls setString( )

GenString(char\* str, int id);

GenString( const GenString &string\_value); // Copy Constructor!!

~GenString(); // Destructor: frees new memory

// \*\* Set and Get Functions

void setString(char \*string\_value); // Sets the string to string\_value

char\* getString() ; // Returns the string

// \*\* Special Functions

// Returns true if string equals another\_string

bool isEqual(GenString another\_string);

void toLower(); // Converts string to lowercase

void toUpper(); // Converts string to uppercase

// Returns a character of the string at the specified index.

// If the index is < 0 or >= length, prints an error message

char getChar( int index);

// Returns the length of string (excluding null)

int getLength() { return strlen(string);}

const char\* getString1();

void change() const;

char& operator[](int a);

};

#endif

-----------------------------------------------------/////---------------------------------------------------------