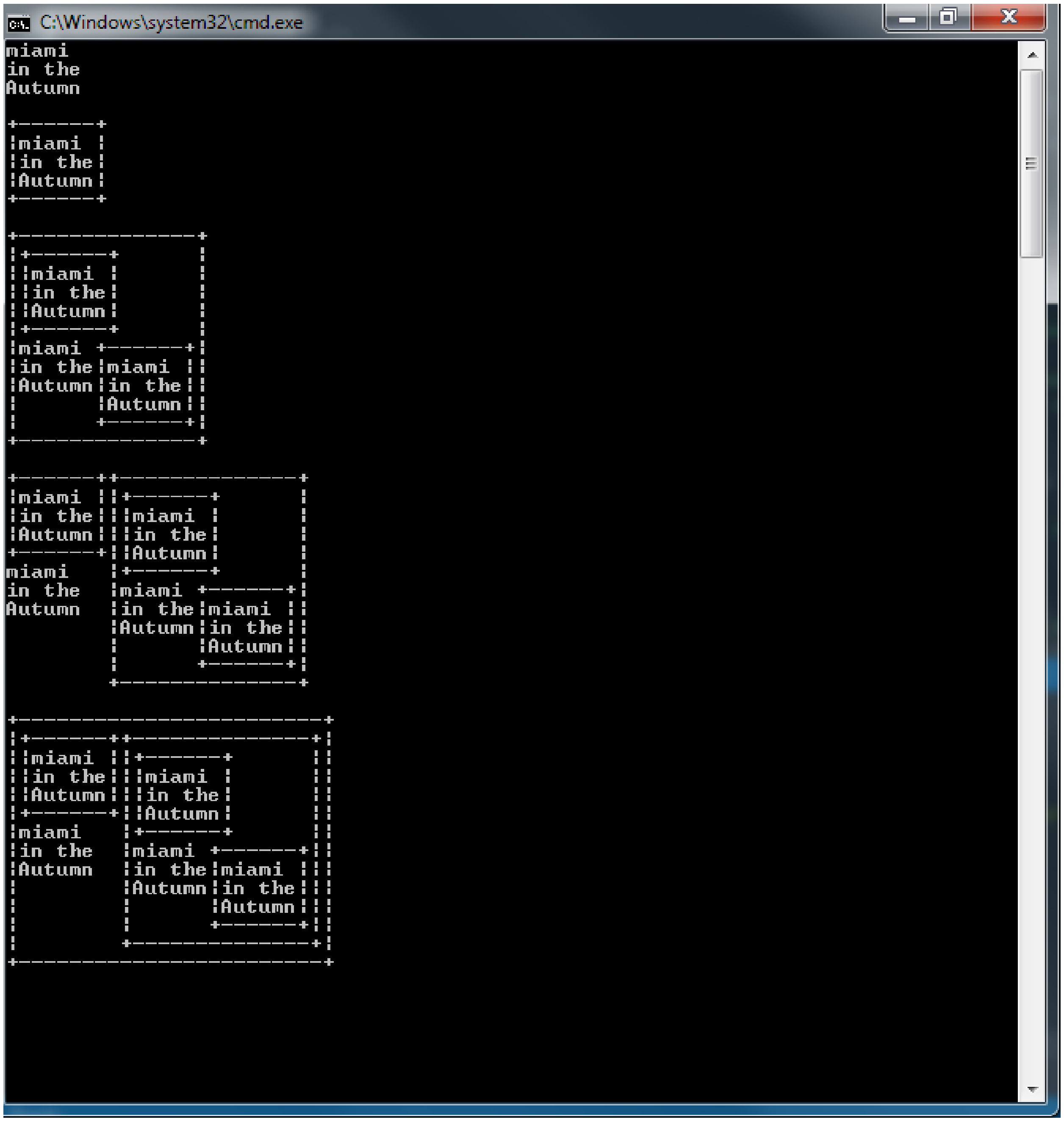
```
//header file Frames.h
#ifndef _Picture_Frames_
#define _Picture_Frames_

#include <iostream>
#include <string>
#include <cstring>

using namespace std;

class Picture{
	//overload the stream extraction operator
	friend ostream& operator <<(ostream& os, const Picture& p);
	friend Picture frame(Picture const& p);
	//bitor operator overload (Bitwise inclusive OR)
	friend Picture operator |(Picture const& p1, Picture const& p2);
	//bitand operator overload (Bitwise AND)
	friend Picture operator &(Picture const& p1, Picture const& p2);
public:
	//default constructor to initialize the string to null
	Picture();
	//constructor
	Picture(int height, int width);
	//constructor; conversion from the char string
	Picture(const char *str[], int size);
	//copy constructor
	Picture(const Picture&);
	//destructor
	~Picture();
	//overload assignment operator
	Picture& operator =(const Picture& p);
private:
	//copies all characters from other (starting at 0,0) to this picture, but starting 
at r, c
	void blockcopy(int row, int col, Picture const& other);
	char& position(int row, int col);
	char position(int row, int col) const;
	static int max(int row, int col);
private:
	//hold the amount of lines the char array will be printed or height of the frame
	int _height;
	//holds the width of the frame
	int _width;
	//pointer to the char array that holds the string/ dynamically alocated array
	char* _data;
};

#endif

/*The class Picture has three private member variables: one to store the array of 
characters and the other
to get the length of the array and another to get the numbers of lines on which the 
character will be printed.
It also contains a private function to copy all characters from other array to
the new picture, but starting at r, c*/
```

```
//Frames.cpp file
#include "Frames.h"//includes the class definition
//overload the stream insertion operator to print the objects from the class
ostream& operator<<(ostream& os, const Picture& p){
	for (int i = 0; i < p._height*p._width; i++)  {//loop to go through the array
		os << p._data[i];//prints the elements in the array
		if (i % p._width == p._width - 1)  { os << endl; }//creates a new line 
after the end of the width is encounter
	}
	return os;
}
//default constructors to store the null string
Picture::Picture() :_height(0), _width(0), _data(nullptr){}
//Constructor that set the spaces in the array to a default character ('space'), and the 
private variables to a default value
Picture::Picture(int height, int width) : _height(height), _width(width), _data(new 
char[_width*_height]){
	memset(_data, ' ', _width*_height);
}
//copies the values from the given array into the class array
Picture::Picture(const char* str[], int size) : _height( size ){
	int  maxlen = 0;//sets the max lenght to 0

	for (int i = 0; i < size; i++)//loop to look for biggest string in the array
		if (maxlen < strlen(str[i]))  { maxlen = strlen(str[i]); }//set the biggest 
string to the max lenght

	_width = maxlen;//sets the max lenght of string to the width of the frame
	_data = new char[_width*_height];//reserves space for the frame
	memset(_data, ' ', _width * _height);//sets all values to a default value '.'

	char* p = _data;//creates a new pointer to the class data variable
	for (int i = 0; i < _height; i++){//loop to go throught the array
		memcpy(p, str[i], strlen(str[i]));//copies all values from the string str 
into the data array pointer p;
		p += _width;//sets every string in the arrray to the max width even if the 
string is less than the width blank
	}                                       //are filled for the rest of spaces if 
string is less then the width.
}
//copy constructor
Picture::Picture(const Picture& p)
:_width(p._width), _height(p._height), _data(new char[p._height*p._width])//sets the 
value from one picture to other
{ blockcopy(0, 0, p); }//calls the block copy to copy the _data from pic1 to pic2
//destructor
Picture::~Picture(){ delete [] _data; }//deleted the class array
//overload assignment operator
Picture& Picture::operator=(const Picture& p){
	if (this != &p){//avoid self-copy
		delete[] _data;//deletes the previous data
		_height = p._height;
		_width = p._width;
		_data = new char[_width*_height];
		memcpy(_data, p._data, _width*_height);//copies the values from data1 to 
another data
	}
	return *this;//returns the  current value being pointed
}
//frames the picture
Picture frame(Picture const& p){
	Picture frame(p._height + 2, p._width + 2);//sets a frame to a bigger frame than
the orignal unframed frame

	memset(frame._data, ' ', frame._width * frame._height);//sets the data in the new
frame to a default space value

	frame.blockcopy(1, 1, p);//copies the data from the original frame to the new
frame
	frame.position(0, 0) = '+';//sets a + sign at each corner of the frame
	frame.position(frame._height - 1, 0) = '+';
	frame.position(0, frame._width - 1) = '+';
	frame.position(frame._height - 1, frame._width - 1) = '+';

	for (int i = 1; i < frame._height - 1; ++i){//creates a | sign at each edge of the
frame
		frame.position(i, 0) = '|';
		frame.position(i, frame._width - 1) = '|';
	}
	memset(frame._data + 1, '-', frame._width - 2);//sets the top and bottom to a -
sign
	memset(frame._data + ((frame._width)*(frame._height - 1))+1, '-', frame._width-2);
	return frame;//returns the a framed picture
}
void Picture::blockcopy(int row, int col, Picture const& other){

	for (int i = 0; i < other._height; ++i){//double for loop to fill the data of the
2d array to the values from the original frame
		for (int j = 0; j < other._width; ++j)
			position(i + row, j + col) = other.position(i, j);
		}
}
char& Picture::position(int row, int col){ return _data[row*_width + col]; }//return the
a position in the 2d array
char  Picture::position(int row, int col)  const  { return _data[row*_width + col]; }
int Picture::max(int row, int col){ return row>col ? row : col; }//returns the max lenght
of the arrray
Picture operator|(Picture const& p1, Picture const& p2){//overload the bitwise inclusive
or operator
	Picture Hcat(Picture::max(p1._height, p2._height), p1._width +
p2._width);//creates a new frame twice the size of the original horizontally
	Hcat.blockcopy(0, 0, p1);//copies the framed frame to a new frame
	Hcat.blockcopy(0, p1._width, p2);
	return Hcat;//returns the horizonatally concatenated framed
}
Picture operator&(Picture const& p1, Picture const& p2){//overload the bitwise and
operator
	Picture Vcat(p1._height + p2._height, Picture::max(p1._width,
p2._width));//creates a new frame to be frame the framed frame
	Vcat.blockcopy(0, 0, p1);//copies the framed frame to a new frame
	Vcat.blockcopy(p1._height, 0, p2);
	return Vcat;//returns the vertically concatenated framed
}
```

```
//main.cpp file
#include <iostream>
#include <string>
#include "Frames.h"//includes the frame class

using namespace std;

const char* init[] = { "miami", "in the", "Autumn"};//given array to be framed

int main(){
	Picture p(init, 3);//creates an unframed pic
	cout << p << endl;//prints out the unframed pic

	Picture p_framed = frame(p);//framed the array
	cout << p_framed << endl;//prints the framed array

	Picture q = frame(p_framed& (p | p_framed));//creates a vertically concatenated 
frame
	cout << q << endl;//prints the frame

	Picture r = (p_framed & p) | q;//creates a horizontallt concatenated frame
	cout << r << endl;//prints the frame

	Picture r_framed = frame(r);//creates a framed frame
	cout << r_framed << endl;//prints out the framed frame

	cin.get();
	return 0;
}

/*SUMMARY
In this project we created use a single class to create a rectangular array of characters 
that can be printed, with and without a frame.
The array used is a dynamic array which stores an array of character pinters, and each 
pointer points to one of the lines in the character picture.
The class named Picture included a costume copy constructor, assignment operator 
overload, and destructor to manage the memeory of the dynamic
array properly. Also in this class 1D pointers were used to manipulate a 2D array.*/
```

```
//Example output
```



C:\Windows\system32\cmd.exe

```
miami
in the
Autumn

+------+
|miami |
|in the|
|Autumn|
+------+

+--------------+
|+------+      |
||miami |      |
||in the|      |
||Autumn|      |
|+------+      |
|miami +------+|
|in the|miami ||
|Autumn|in the||
|      |Autumn||
|      +------+|
+--------------+

+------++--------------+
|miami ||+------+      |
|in the|||miami |      |
|Autumn|||in the|      |
+------+||Autumn|      |
miami   |+------+      |
in the  |miami +------+|
Autumn  |in the|miami ||
        |Autumn|in the||
        |      |Autumn||
        |      +------+|
        +--------------+

+------------------------+
|+------++--------------+|
||miami |||+------+      ||
||in the||||miami |      ||
||Autumn||||in the|      ||
|+------+|||Autumn|      ||
|miami   ||+------+      ||
|in the  ||miami +------+||
|Autumn  ||in the|miami |||
|        ||Autumn|in the|||
|        ||      |Autumn|||
|        ||      +------+||
|        |+--------------+|
+------------------------+
```