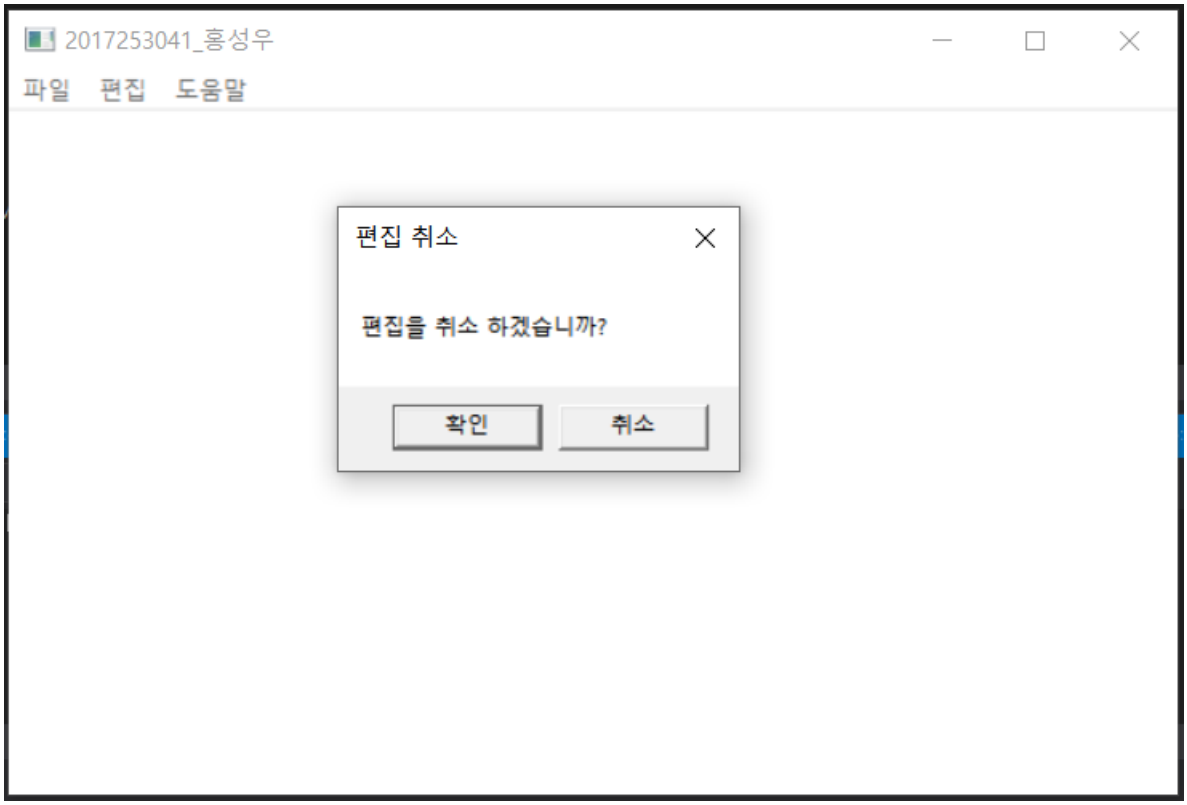
< 윈도우프로그래밍 5주차 과제 >

2017253041_홍성우

#1.

```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
	WPARAM wParam, LPARAM lParam);
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
	LPSTR lpszCmdLine, int nCmdShow)
	// UNICODE 사용시 wWinMain() 형태
	// hPrevInstance 이전 인스턴스 항상 0값
	// lpszCmdLine > 외부에서 (내부로) 입력받는 변수
	// nCmdShow 윈도우 출력 형태에 관련한 값
{
	HWND	hwnd;
	MSG		msg;
	WNDCLASS WndClass;
	WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
	WndClass.lpfnWndProc = WndProc;			// Proc 설정
	WndClass.cbClsExtra = 0;
	WndClass.cbWndExtra = 0;
	WndClass.hInstance = hInstance;
	WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
	WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
	WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);	// 형변환
	WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5_1);
	WndClass.lpszClassName = _T("Window Class Name");
	RegisterClass(&WndClass);		// WndClass 등록
	hwnd = CreateWindow(_T("Window Class Name"),
		_T("2017253041_홍성우"),		// 타이틀바, 학번이름 체크
		WS_OVERLAPPEDWINDOW,		// 윈도우 스타일
		600, 400,					// 창출력좌표 x, y
		600, 400,					// 창크기 x, y축
		NULL,						// 부모 윈도우
		NULL,						// 메뉴바 핸들
		hInstance,					// 인스턴스
		NULL						// 여분, NULL
	);
	ShowWindow(hwnd, nCmdShow);		// 윈도우 출력, WM_PAINT 출력내용 가져옴
	UpdateWindow(hwnd);				// WM_PAINT 출력내용 발생해서
출력하도록
					// hwnd 핸들을 통해
보여주고 갱신

	//ShowWindow(hwnd, SW_SHOW);	// 위와 같음
	//UpdateWindow(hwnd);

	while (GetMessage(&msg, NULL, 0, 0))	// 메시지 큐의 메시지를 가져옴
	{
		TranslateMessage(&msg);		// 키입력에 반응하는 메시지 변환, WM_KEYDOWN
```

```
(키가 눌릴때) WM_CHAR 메시지 발생
                DispatchMessage(&msg);          // WndProc() 함수 호출과 WndProc()으로
메세지 전달
        }                                                               // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
        return (int)msg.wParam;                 // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam)
{
        int answer;

        switch (iMsg)
        {
        case WM_CREATE:
                break;
        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case ID_EDITUNDO:
                        answer = MessageBox(hwnd,
                                _T("편집을 취소 하겠습니까?"),
                                _T("편집 취소"),
                                MB_OKCANCEL);
                        break;
                }
                break;
        case WM_PAINT:
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        }
        return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```

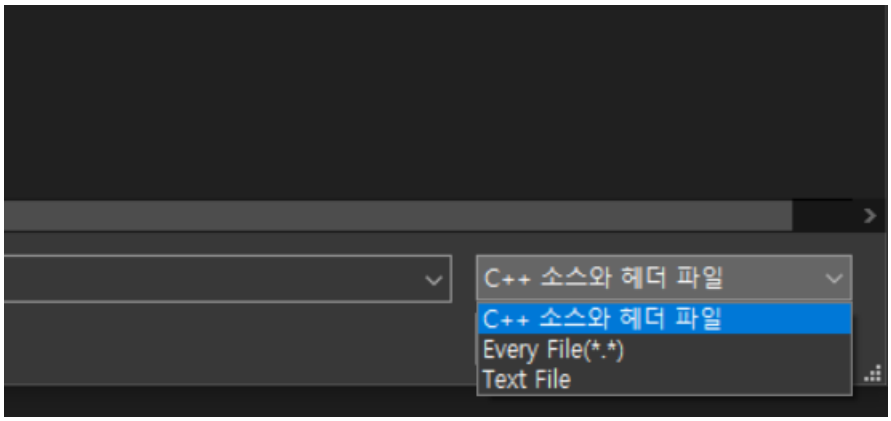


#2.

```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpszCmdLine, int nCmdShow)
        // UNICODE 사용시 wWinMain() 형태
        // hPrevInstance 이전 인스턴스 항상 0값
        // lpszCmdLine > 외부에서 (내부로) 입력받는 변수
        // nCmdShow 윈도우 출력 형태에 관련한 값
{
        HWND     hwnd;
        MSG              msg;
        WNDCLASS WndClass;
        WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
        WndClass.lpfnWndProc = WndProc;           // Proc 설정
        WndClass.cbClsExtra = 0;
        WndClass.cbWndExtra = 0;
        WndClass.hInstance = hInstance;
        WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
        WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
        WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);      // 형변환
        WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5_1);
        WndClass.lpszClassName = _T("Window Class Name");
        RegisterClass(&WndClass);         // WndClass 등록
        hwnd = CreateWindow(_T("Window Class Name"),
                _T("2017253041_홍성우"),          // 타이틀바, 학번이름 체크
```

```
                WS_OVERLAPPEDWINDOW,            // 윈도우 스타일
                600, 400,                                       // 창출력좌표 x, y
                600, 400,                                       // 창크기 x, y축
                NULL,                                           // 부모 윈도우
                NULL,                                           // 메뉴바 핸들
                hInstance,                                      // 인스턴스
                NULL                                            // 여분, NULL
        );
        ShowWindow(hwnd, nCmdShow);             // 윈도우 출력, WM_PAINT 출력내용 가져옴
        UpdateWindow(hwnd);                             // WM_PAINT 출력내용 발생해서
출력하도록
                                                                        // hwnd 핸들을 통해
보여주고 갱신

        //ShowWindow(hwnd, SW_SHOW);    // 위와 같음
        //UpdateWindow(hwnd);

        while (GetMessage(&msg, NULL, 0, 0))    // 메시지 큐의 메시지를 가져옴
        {
                TranslateMessage(&msg);         // 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
                DispatchMessage(&msg);          // WndProc() 함수 호출과 WndProc()으로
메세지 전달
        }                                                       // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
        return (int)msg.wParam;                 // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam)
{
        OPENFILENAME OFN;
        TCHAR str[100], lpstrFile[100] = _T("");
        TCHAR filter[] = _T("C++ 소스와 헤더 파일 ₩0 *.cpp ₩0Every File(*.*) ₩0 *.* ₩0Text
File ₩0 *.txt ; *.doc₩0");

        switch (iMsg)
        {
        case WM_CREATE:
                break;
        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case ID_FILEOPEN:
                        memset(&OFN, 0, sizeof(OPENFILENAME));
                        OFN.lStructSize = sizeof(OPENFILENAME);
                        OFN.hwndOwner = hwnd;
                        OFN.lpstrFilter = filter;
                        OFN.lpstrFile = lpstrFile;
                        OFN.nMaxFile = 100;
                        OFN.lpstrInitialDir = _T(".");
                        if (GetOpenFileName(&OFN) != 0)
                        {
```

```
                                _stprintf_s(str, _T("%s 파일을 열겠습니까?"), OFN.lpstrFile);
                                MessageBox(hwnd, str, _T("열기 선택"), MB_OK);
                        }
                        break;
                }
                break;
        case WM_PAINT:
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        }
        return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```



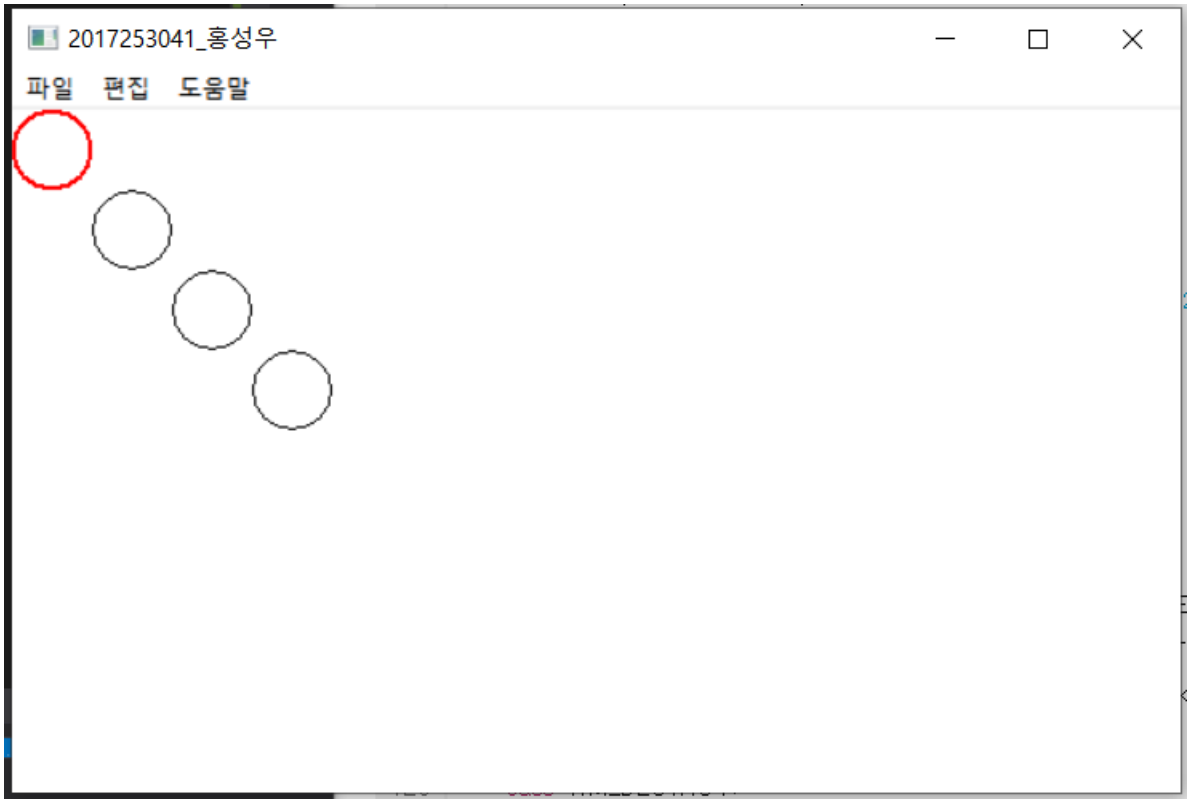
#3.

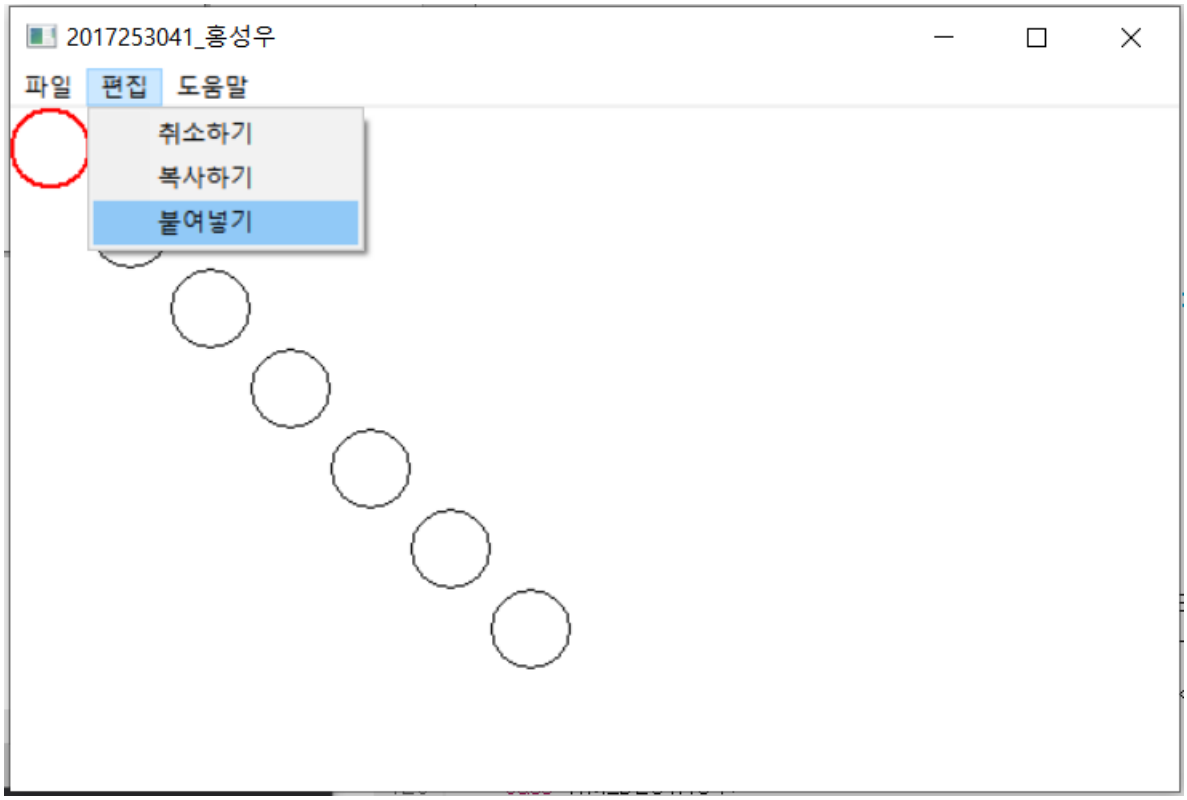
```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpszCmdLine, int nCmdShow)
        // UNICODE 사용시 wWinMain() 형태
        // hPrevInstance 이전 인스턴스 항상 0값
        // lpszCmdLine > 외부에서 (내부로) 입력받는 변수
        // nCmdShow 윈도우 출력 형태에 관련한 값
{
        HWND     hwnd;
        MSG              msg;
        WNDCLASS WndClass;
        WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
        WndClass.lpfnWndProc = WndProc;           // Proc 설정
        WndClass.cbClsExtra = 0;
        WndClass.cbWndExtra = 0;
        WndClass.hInstance = hInstance;
        WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
        WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
        WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);       // 형변환
```

```
	WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5_1);
	WndClass.lpszClassName = _T("Window Class Name");
	RegisterClass(&WndClass);		// WndClass 등록
	hwnd = CreateWindow(_T("Window Class Name"),
		_T("2017253041_홍성우"),		// 타이틀바, 학번이름 체크
		WS_OVERLAPPEDWINDOW,		// 윈도우 스타일
		600, 400,					// 창출력좌표 x, y
		600, 400,					// 창크기 x, y축
		NULL,						// 부모 윈도우
		NULL,						// 메뉴바 핸들
		hInstance,					// 인스턴스
		NULL						// 여분, NULL
	);
	ShowWindow(hwnd, nCmdShow);		// 윈도우 출력, WM_PAINT 출력내용 가져옴
	UpdateWindow(hwnd);				// WM_PAINT 출력내용 발생해서
출력하도록
                                                        // hwnd 핸들을 통해
보여주고 갱신

	//ShowWindow(hwnd, SW_SHOW);	// 위와 같음
	//UpdateWindow(hwnd);

	while (GetMessage(&msg, NULL, 0, 0))	// 메시지 큐의 메시지를 가져옴
	{
		TranslateMessage(&msg);		// 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
		DispatchMessage(&msg);		// WndProc() 함수 호출과 WndProc()으로
메세지 전달
	}                                                   // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
	return (int)msg.wParam;			// wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}

#include <math.h>
#define BSIZE 20
int x[10] = { 20,60,100,140,180,220,260,300,340,380 };
int y[10] = { 20,60,100,140,180,220,260,300,340,380 };
static int count = 0;
float LengthPts(int x1, int y1, int x2, int y2)
{
	return(sqrt((float)((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1))));
}
BOOL InCircle(int x, int y, int mx, int my)
{
	if (LengthPts(x, y, mx, my) < BSIZE) return TRUE;
	else return FALSE;
}
int SelectCircle(int mx, int my)
{
	for (int i = 0; i < count; i++)
		if (InCircle(x[i], y[i], mx, my))
			return i;
	return -1;
```

```
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
	WPARAM wParam, LPARAM lParam)
{
	HDC hdc;
	PAINTSTRUCT ps;
	int mx, my;
	HPEN hPen, oldPen;
	static bool Selection;

	switch (iMsg)
	{
	case WM_CREATE:
		Selection = false;
		break;
	case WM_LBUTTONDOWN:
		mx = LOWORD(lParam);
		my = HIWORD(lParam);
		for (int i = 0; i <= count; i++)
		{
			if (InCircle(x[i], y[i], mx, my))
			{
				if (count < 9) {
					count++;
					Selection = true;
					InvalidateRgn(hwnd, NULL, TRUE);
					break;
				}
				else Selection = false;
			}
			else Selection = false;
		}
		InvalidateRgn(hwnd, NULL, TRUE);
		break;
	case WM_COMMAND:
		switch (LOWORD(wParam))
		{
		case ID_EDITPASTE:
			if (count < 9)
			{
				count++;
				Selection = true;
			}
			else Selection = false;
			InvalidateRgn(hwnd, NULL, TRUE);
			break;
		}
		break;
	case WM_PAINT:
		hdc = BeginPaint(hwnd, &ps);
		hPen = (HPEN)CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
		oldPen = (HPEN)SelectObject(hdc, hPen);
		Ellipse(hdc, x[0] - 20, y[0] - 20, x[0] + 20, y[0] + 20);
		SelectObject(hdc, oldPen);
```

```
                DeleteObject(hPen);
                if (Selection)
                {
                        for (int j = 0; j < count; j++)
                        {
                                Ellipse(hdc, x[j + 1] - 20, y[j + 1] - 20, x[j + 1] + 20, y[j
+ 1] + 20);
                        }
                }
                EndPaint(hwnd, &ps);
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        }
        return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```





#4.

```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
	WPARAM wParam, LPARAM lParam);
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
	LPSTR lpszCmdLine, int nCmdShow)
	// UNICODE 사용시 wWinMain() 형태
	// hPrevInstance 이전 인스턴스 항상 0값
	// lpszCmdLine > 외부에서 (내부로) 입력받는 변수
	// nCmdShow 윈도우 출력 형태에 관련한 값
{
	HWND	hwnd;
	MSG		msg;
	WNDCLASS WndClass;
	WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
	WndClass.lpfnWndProc = WndProc;		// Proc 설정
	WndClass.cbClsExtra = 0;
	WndClass.cbWndExtra = 0;
	WndClass.hInstance = hInstance;
	WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
	WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
	WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);	// 형변환
	WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5_1);
	WndClass.lpszClassName = _T("Window Class Name");
	RegisterClass(&WndClass);		// WndClass 등록
	hwnd = CreateWindow(_T("Window Class Name"),
		_T("2017253041_홍성우"),		// 타이틀바, 학번이름 체크
```

```
                WS_OVERLAPPEDWINDOW,            // 윈도우 스타일
                600, 400,                                       // 창출력좌표 x, y
                600, 400,                                       // 창크기 x, y축
                NULL,                                           // 부모 윈도우
                NULL,                                           // 메뉴바 핸들
                hInstance,                                      // 인스턴스
                NULL                                            // 여분, NULL
        );
        ShowWindow(hwnd, nCmdShow);             // 윈도우 출력, WM_PAINT 출력내용 가져옴
        UpdateWindow(hwnd);                             // WM_PAINT 출력내용 발생해서
출력하도록
                                                                        // hwnd 핸들을 통해
보여주고 갱신

        //ShowWindow(hwnd, SW_SHOW);    // 위와 같음
        //UpdateWindow(hwnd);

        while (GetMessage(&msg, NULL, 0, 0))    // 메시지 큐의 메시지를 가져옴
        {
                TranslateMessage(&msg);         // 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
                DispatchMessage(&msg);          // WndProc() 함수 호출과 WndProc()으로
메세지 전달
        }                                                       // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
        return (int)msg.wParam;                 // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam)
{
        HDC hdc;
        PAINTSTRUCT ps;
        int mx, my;
        HPEN hPen, oldPen;
        HBRUSH hBrush, oldBrush;
        int endX = 0, endY = 0;
        static bool Drag;

        switch (iMsg)
        {
        case WM_CREATE:
                break;
        case WM_MOUSEMOVE:
                hdc = GetDC(hwnd);
                if (Drag)
                {
                        SetROP2(hdc, R2_XORPEN);
                        oldPen = (HPEN)SelectObject(hdc, hPen);
                        oldBrush = (HBRUSH)SelectObject(hdc, hBrush);
                        endX = LOWORD(lParam);
                        endY = HIWORD(lParam);
                        switch (object_mode)
```

```
                        {
                        case LINE:
                                MoveToEx(hdc, startX, startY, NULL);
                                LineTo(hdc, oldX, oldY);
                                MoveToEx(hdc, startX, startY, NULL);
                                LineTo(hdc, endX, endY);
                                break;
                        case ELLIPSE:
                                Ellipse(hdc, startX, startY, oldX, oldY);
                                Ellipse(hdc, startX, startY, endX, endY);
                                break;
                        case RECTANGLE:
                                Rectangle(hdc, startX, startY, oldX, oldY);
                                Rectangle(hdc, startX, startY, endX, endY);
                                break;
                        }
                        oldX = endX; oldY = endY;
                        SelectObject(hdc, oldPen);
                        SelectObject(hdc, oldBrush);
                }
                ReleaseDC(hwnd, hdc);
                break;
        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case ID_LINE:
                        object_mode = LINE;
                        break;
                case ID_ELLIPSE:
                        object_mode = ELLIPSE;
                        break;
                case ID_RECTANGLE:
                        object_mode = RECTANGLE;
                        break;
                case ID_PENCOLOR:
                        color_mode = PEN;
                        hPen = CreatePen(PS_SOLID, 1, ColorSelection(hwnd,
                                color_mode));
                        break;
                case ID_FACECOLOR:
                        color_mode = BRUSH;
                        hBrush = CreateSolidBrush(ColorSelection(hwnd, color_mode));
                        break;
                }
                break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        }
        return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```

#5.

```
#include <windows.h>
#include <TCHAR.H>
#include "resource.h"
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam);
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
        LPSTR lpszCmdLine, int nCmdShow)
        // UNICODE 사용시 wWinMain() 형태
        // hPrevInstance 이전 인스턴스 항상 0값
        // lpszCmdLine > 외부에서 (내부로) 입력받는 변수
        // nCmdShow 윈도우 출력 형태에 관련한 값
{
        HWND     hwnd;
        MSG               msg;
        WNDCLASS WndClass;
        WndClass.style = CS_HREDRAW | CS_VREDRAW; //height, vertical redraw
        WndClass.lpfnWndProc = WndProc;           // Proc 설정
        WndClass.cbClsExtra = 0;
        WndClass.cbWndExtra = 0;
        WndClass.hInstance = hInstance;
        WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
        WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
        WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);      // 형변환
        WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU5_1);
        WndClass.lpszClassName = _T("Window Class Name");
        RegisterClass(&WndClass);         // WndClass 등록
        hwnd = CreateWindow(_T("Window Class Name"),
                _T("2017253041_홍성우"),          // 타이틀바, 학번이름 체크
                WS_OVERLAPPEDWINDOW,              // 윈도우 스타일
                600, 400,                                   // 창출력좌표 x, y
                600, 400,                                   // 창크기 x, y축
                NULL,                                       // 부모 윈도우
                NULL,                                       // 메뉴바 핸들
                hInstance,                                  // 인스턴스
                NULL                                        // 여분, NULL
        );
        ShowWindow(hwnd, nCmdShow);               // 윈도우 출력, WM_PAINT 출력내용 가져옴
        UpdateWindow(hwnd);                               // WM_PAINT 출력내용 발생해서
출력하도록
                                                                  // hwnd 핸들을 통해
보여주고 갱신

        //ShowWindow(hwnd, SW_SHOW);      // 위와 같음
        //UpdateWindow(hwnd);

        while (GetMessage(&msg, NULL, 0, 0))      // 메시지 큐의 메시지를 가져옴
        {
                TranslateMessage(&msg);           // 키입력에 반응하는 메시지 변환, WM_KEYDOWN
(키가 눌릴때) WM_CHAR 메시지 발생
                DispatchMessage(&msg);            // WndProc() 함수 호출과 WndProc()으로
메세지 전달
        }                                                 // 종료는 WM_QUIT
발생할때 FALSE 리턴하면서 종료
```

```
        return (int)msg.wParam;                    // wParam, lParam 윈도우 크기가 어떻게
변했는지, 변경된 클라이언트, 키보드, 마우스 값

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam)
{
        HDC hdc;
        PAINTSTRUCT ps;
        int mx, my;
        HPEN hPen, oldPen;
        HBRUSH hBrush, oldBrush;
        int endX = 0, endY = 0;
        static bool Drag;

        switch (iMsg)
        {
        case WM_CREATE:
                break;
        case VK_BACK:
                Rectangle(hdc, left, top, right, bottom);
                if (count > 0)
                        count--;
                else
                        if (line > 0)
                        {
                                line = line - 1;
                                count = _tcslen(str[line]);
                        }
                break;
        case VK_RETURN:
                count = 0;
                line = line + 1;
                break;
        default:
                str[line][count++] = wParam;
        }
        GetTextExtentPoint(hdc, str[line], _tcslen(str[line]), &size);
        if (size.cx > right - left)
        {
                str[line + 1][0] = str[line][count - 1];
                str[line][count - 1] = NULL;
                line++;
                count = 1;
        }
        str[line][count] = NULL;
        if ((line + 1) * 20 > bottom - top)
        {
                MessageBox(hwnd, _T("글상자가 가득 찼습니다."), _T("글상자"), MB_OK);
                line = line - 1;
                count = _tcslen(str[line]);
                return;
        }
        for (i = 0; i < line; i++)
```

```
                TextOut(hdc, left + 0, top + i * 20, str[i], _tcslen(str[i]));
        TextOut(hdc, left + 0, top + line * 20, str[line], _tcslen(str[line]));
        return;
}
case WM_CREATE:
        object_mode = 0;
        color_mode = 0;
        Drag = FALSE;
        hPen = (HPEN)GetStockObject(WHITE_PEN);
        hBrush = (HBRUSH)GetStockObject(BLACK_BRUSH);
        TextBoxOn = FALSE;
        break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        }
        return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```