단국대학교 멀티미디어공학 HCI 프로그래밍 II 중간고사 (2008년 가을학기) 2008년 10월 22일

학과 _____ 학번 _____ 이름 _____

중간고사

담당교수: 단국대학교 멀티미디어공학전공 박경신

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안 함. 답에는 반드시 네모를 쳐서 확실히 표시할 것.
- 답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번 대신 암호를 사용할 것임.

1. 다음 프로그램에서 실행되는 출력결과를 써라. (10점)

```
#include <iostream>
using namespace std;

void add1(int d, int e, int &f);
void add2(void);

int i = 5;
int a = 3, b = 4, c = 6;
int * ptr;

int main()
{
   ptr = &i;
   ++i;
   cout << "main : i = " << i << endl;
   cout << "main : a = " << a << " b = " << b << " c = " << c << endl;
   add1(a, b, c);
   cout << "main : i = " << i << endl;
   *ptr = 10;
   cout << “main : i = “ << i << endl;
   return 0;
}

void add1(int d, int e, int &f)
{
   int i = 4;

   ++i;
   cout << "add1 : i = " << i << endl;
   add2();
   f = d + e;
   cout << "add1 : a = " << a << " b = " << b << " c = " << c << endl;
}

void add2(void)
{
   ++i;
   cout << "add2 : i = " << i << endl;
}
```

```
main : i = 6
main : a = 3 b = 4 c = 6
add1 : i = 5
add2 : i = 7
add1 : a = 3 b = 4 c = 7
main : i = 7
main : i = 10
```

학과 _____ 학번 _____ 이름 _____

2. 다음은 루프(for, while loop)를 사용해서 완성된 factorial 함수이다. 재귀호출(recursive call)을 이용해서 완성하시오. (10점)

```
long factorial (long a)
{
        long result = 1;
        for ( long i = a; i > 0; i-- ) {
                result *= i;
        }
        return result;
}
```

```
long factorial (long a)
{
        if (a > 1)
                return (a * factorial(a-1));
        else
                return 1;

}
```

3. 함수 오버로딩 (function overloading), 함수 오버라이딩 (function overriding), 연산자 오버로딩 (operator overloading)을 예를 들어서 간단히 설명하라. (10점)

- 함수 오버로딩 (function overloading)
  - 동일한 함수명에 매개변수가 다른 함수를 둘 이상 정의하는 것으로, 동일한 함수 기능을 수행하지만 다른 매개변수의 경우를 처리할 때 사용

```
int Max(int a, int b);
double Max(double a, double b);
```

- 함수 오버라이딩 (function overriding)
  - 상속받은 파생 클래스에서 동일한 함수명에 동일한 매개변수로 정의하여 함수를 재정의하는 것으로, 상속되어진 함수의 기능을 변경해서 재사용하고 싶을 때 사용

```
class Point { void Show(); }                    // show()함수가 x,y를 출력
class Point3D : public Point{ void Show(); }    // show()함수가 x,y,z를 출력
```

- 연산자 오버로딩 (operator overloading)
  - 클래스에서 연산자를 함수처럼 새롭게 재정의하여 사용하는 것으로, ==, [], + 연산자 등을 오버로딩함으로써 프로그램의 가독성을 높일 수 있음

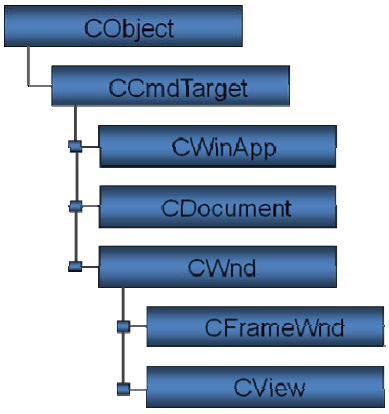
```
class Vector {
        Vector& operator=(const Vector&);
        Vector& operator+(const Vector&);
        bool operator==(const Vector&) const;
        friend ostream& operator<<(ostream&, const Vector&);
};
```

학과 _____ 학번 _____ 이름 _____

4. CObject 클래스가 제공하는 대표적인 서비스 4가지인 실시간 클래스 정보(run-time class information), 동적 객체 생성 (dynamic object creation), 직렬화 (serialization), 실시간 객체 상태검사(run-time object diagnostics)에 대해 간단히 설명하고, 관련함수와 매크로를 적어라. (10점)

- 실시간 클래스 정보 (run-time class information)
  프로그램 실행 (run-time) 중에 객체의 정보 (클래스 타입, 크기 등)를 알 수 있게 해줌. DECLARE_DYNAMIC/IMPLEMENT_DYNAMIC 매크로를 추가하여 IsKindOf를 호출할 수 있게 됨. GetRuntimeClass는 객체의 클래스와 일치하는 CRuntimeClass 구조체를 반환. IsKindOf는 주어진 클래스와 객체와의 관계를 검사.
- 동적 객체 생성 (dynamic object creation)
  개발자가 정의한 CObject에서 파생된 클래스에 동적으로 객체를 생성할 수 있게 해줌. DECLARE_DYNCREATE/IMPLEMENT_DYNCREATE 매크로를 추가하면 동적 객체 생성 사용 가능.
- 직렬화 (serialization)
  객체의 상태를 저장하거나 읽어 들일 수 있게 해줌. DEBLARE_SERIAL/IMPLEMENT_SERIAL 매크로를 추가하고, 개발자가 정의한 멤버 변수를 read/write하는 Serialize()를 재정의(overriding)해야 함. IsSerializable은 객체가 직렬화될 수 있는 지를 알아보기 위해 검사. Serialize는 Archive를 이용해 일반 파일에 객체를 저장하거나 또는 파일로부터 읽어들임.
- 실시간 객체 상태 검사 (run-time object diagnostics)
  객체 상태를 점검하게 해줌. AssertValid 는 실시간으로 객체의 멤버들에 대한 유효성 여부를 검사. Dump는 객체의 멤버들에 대한 진단 덤프를 생성.

5. 다음은 MFC 응용프로그램의 기본 구조를 위한 클래스인 AFX Class를 보여주고 있다. 각 클래스의 기능을 간단히 설명하라. (10점)



CObject – MFC 클래스의 기반 클래스로 대부분의 MFC 클래스가 CObject를 상속받음

CCmdTarget – 메시지 맵 구조 지원, WM_COMMAND 메시지 처리

CWinApp – 응용 프로그램 관리, 구동 클래스로 인스턴스를 초기화하고 메시지 루프를 설정. 프로그램의 시작과 종료를 담당, 프로그램이 시작될 때 메인 프레임 윈도우 생성, 메시지 루프 처리, WM_QUIT 메시지가 들어오면 루프를 빠져나와 종료함, 프로그램을 전체를 대표하는 기능들 수행

CDocument – 프로그램에서 사용되는 데이터 구조, 파일을 관리(읽기, 저장, 새로운 파일 생성, 파일 닫기, 변경된 데이터를 뷰 객체에 알려주는 기능)

CWnd – 화면에 보여지는 모든 윈도우 (프레임 창, 대화상자, 자식 창, 컨트롤 및 도구모음 등) 대한 기반 클래스로 윈도우 관리에 필요한 기능 관리

CFrameWnd – 윈도우 이동, 크기 조절 등 윈도우의 프레임을 관리

CView – 윈도우의 클라이언트 영역을 관리하는 클래스로 데이터를 화면에 보여주는 기능, 클라이언트

영역에서 발생되는 마우스, 키보드 메시지 처리를 위한 메시지 핸들러 오버라이딩

6. MFC의 디바이스 컨텍스트 그리기 클래스를 적고, 간단히 설명하라. (5점)

CPaintDC – OnPaint함수에서 사용하는 화면출력 DC 관리

CClientDC – OnPaint() 함수 외에서 사용되는 클라이언트 영역에 화면출력 DC 관리

CWindowDC – 윈도우 전체영역 (비클라이언트+클라이언트)에 그래픽 출력 DC 관리

CMetaFileDC – GDI 명령어를 저장할 수 있는 파일

7. MFC의 GDI 객체 클래스를 5개 이상 적고, 어떻게 사용하는지 간단히 설명하라. (5점)

CGdiObject – Cbitmap, CBrush, CFont, CPalette, CPen, CRgn 등의 기반클래스

CBitmap – 비트맵 그림을 다룰 때 사용

CBrush – 면의 내부를 채울 때 사용

CFont – 문자출력을 위한 글자의 모양, 크기를 설정할 때 사용

CPalette – 출력될 색의 집합

CPen – 선을 그릴 때 사용

CRgn – 다양한 형태의 면을 정의할 때 사용

8. Win32 SDK 윈도우 응용프로그램과 MFC 응용프로그램의 "출력방법"을 HelloSDK와 HelloMFC 프로그램의 예를 들어 비교 서술하라. (5점)

SDK 프로그램은 윈도우 운영체제에게 DC를 요청하고, 운영체제로부터 받은 DC 핸들을 사용하여 출력하고, 출력 후에 운영체제에게 DC 사용이 끝났음을 알림.

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch(message) {
// 중간 생략…
    case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);                    // DC를 얻음
        TextOut(hdc, 100, 100, str, lstrlen(str));      // HDC를 넣고, 출력함수 호출
        EndPaint(hwnd, &ps);                            // DC를 반환
        return 0;
// 중간 생략…
}
```

MFC는 DC 클래스 객체를 생성하고, DC 객체의 멤버함수를 호출하여 출력함.

```
void CMainFrame::OnPaint()
{
```

```
    char *msg = "Hello, MFC";
    CPaintDC dc(this);      // CPaintDC에서 내부적으로 BeginPaint/EndPaint 호출
    dc.TextOut(100, 100, msg, lstrlen(msg));    // dc의 출력함수 호출
}
```

9. Win32 SDK 윈도우 응용프로그램과 MFC 응용프로그램의 "메시지 처리방법"에 대하여 HelloSDK와 HelloMFC 프로그램의 예를 들어 비교 서술하라. (5점)

SDK 프로그램은 응용프로그램으로 들어온 메시지를 WinProc() 함수 내에서 switch 구문을 사용하여 처리함

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch(message) {
    case WM_CREATE:  // 윈도우가 생성됐을 때
        return 0;
    case WM_PAINT:          // 화면에 출력하는 메시지
        hdc = BeginPaint(hwnd, &ps);
        TextOut(hdc, 100, 100, str, lstrlen(str));
        EndPaint(hwnd, &ps);
        return 0;
    case WM_LBUTTONDOWN:    // 왼쪽 마우스 버튼이 눌렸을 때 발생되는 메시지
        MessageBox(hwnd, "마우스를 클릭했습니다.", "마우스 메시지", MB_OK);
        return 0;
// 중간 생략…
}
```

MFC 프로그램은 메시지 처리를 위해 메시지맵(Message map)을 사용하고 메시지 핸들러 함수를 구현하여 사용함. 메시지맵이란 메시지가 발생했을 때 호출되는 함수의 포인터 등의 정보를 갖고 있는 테이블로 프로그램에 전달된 메시지와 메시지 핸들러 함수를 연결하는데 사용함. 파생 클래스의 메시지 핸들러 함수가 우선적으로 호출됨.

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_PAINT()
    ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()

void CMainFrame::OnPaint()
{
    char *msg = "Hello, MFC";
```

```
    CPaintDC dc(this);
    dc.TextOut(100, 100, msg, lstrlen(msg));
}

void CMainFrame::OnLButtonDown(UINT nFlags, CPoint point)
{
    MessageBox("마우스를 클릭했습니다.", "마우스 메시지");
}
```

10. 다음은 Application Wizard로 생성한 Simple MFC 응용프로그램의 일부를 보여주고 있다. 밑줄 친 부분의 CSimpleApp, CMainFrame, CChildView 클래스 주요 함수와 역할을 간단히 설명하라. 그리고 이 프로그램 소스코드에서 보이지 않는 CSimpleApp의 기반 클래스인 CWinApp의 Run() 함수의 역할을 간단히 설명하라. (10점)

```
// 응용프로그램의 메시지 루프를 수행, InitInstance() 호출 후에 호출되어
// 응용프로그램에 들어온 메시지를 분석하여 메시지 핸들러에 보내어 처리를 하게 한다
virtual int CWinApp::Run()
```

```
------ Simple.cpp : Defines the class behaviors for the application------
#include "stdafx.h"
#include "Simple.h"
#include "MainFrm.h"

// 중간 생략….
```

```
// 응용 프로그램이 시작되면 WinMain() 함수가 호출되어
// CWinApp로부터 파생된 클래스인 CSimpleApp 클래스의 전역 객체 theApp을 찾는다
CSimpleApp theApp;
```

```
// CSimpleApp의 InitInstance() 함수가 호출되어 프레임 윈도우 (pFrame)가 생성되고
// 화면에 출력(ShowWindow & UpdateWindow)된다
BOOL CSimpleApp::InitInstance()
{
        // 중간 생략….

        // Standard initialization
        SetRegistryKey(_T("Local AppWizard-Generated Applications"));
        CMainFrame* pFrame = new CMainFrame;
```

```
	if (!pFrame)
		return FALSE;
	m_pMainWnd = pFrame;
	// create and load the frame with its resources
	pFrame->LoadFrame(IDR_MAINFRAME,
		WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, NULL,
		NULL);
	// The one and only window has been initialized, so show and update it
	pFrame->ShowWindow(SW_SHOW);
	pFrame->UpdateWindow();

	return TRUE;
}
// 중간 생략….
```

------ MainFrm.cpp : implementation of the CMainFrame class------

```
#include "stdafx.h"
#include "Simple.h"
#include "MainFrm.h"

// CMainFrame
IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
	ON_WM_CREATE()
	ON_WM_SETFOCUS()
END_MESSAGE_MAP()

// 중간 생략….
```

// WM_CREATE 메시지를 받아 CMainFrame의 OnCreate() 함수가 호출되어,
// 뷰가 생성(m_wndView.Craete(…))된다

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;
	// create a view to occupy the client area of the frame
	if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
```

```
            CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL)) {
            TRACE0("Failed to create view window\n");
            return -1;
      }

      // 중간 생략….

      return 0;
}

// 중간 생략….
```

// WM_SETFOCUS 메시지를 받아 뷰 (m_wndView)의 SetFocus를 부른다

```
void CMainFrame::OnSetFocus(CWnd* /*pOldWnd*/)
{
      // forward focus to the view window
      m_wndView.SetFocus();
}
// 중간 생략….
```

```
------ ChildView.cpp : implementation of the CChildView class------
#include "stdafx.h"
#include "Simple.h"
#include "ChildView.h"

// 중간 생략….

BEGIN_MESSAGE_MAP(CChildView, CWnd)
      ON_WM_PAINT()
END_MESSAGE_MAP()

// 중간 생략….
```

// WM_PAINT 메시지를 받아 클라이언트 영역에 "Simple 안녕하세요"를 출력한다

```
void CChildView::OnPaint()
{
      CPaintDC dc(this);
      dc.TextOut(100, 100, CString("Simple 안녕하세요."));
```

```
}
```

11. 다음은 간단한 MFC 프로그램 예를 보여주고 있다. 이 프로그램의 전체적인 실행 결과를 그림으로 그리고 간단하게 서술하라. (10점)

```
------ ChildView.cpp : implementation of the CChildView class------
#include "stdafx.h"
#include "SimpleDrawMsg.h" // 메인응용프로그램
#include "ChildView.h"
// 중간 생략….

BEGIN_MESSAGE_MAP(CChildView, CWnd)
	ON_WM_PAINT()
	ON_WM_LBUTTONDOWN()
	ON_WM_CHAR()
END_MESSAGE_MAP()
// 중간 생략….

void CChildView::OnPaint()
{
	CPaintDC dc(this); // device context for painting

	// TODO: Add your message handler code here
	CPen bdpen(PS_DASHDOT, 1, RGB(0, 0, 255)), *oldpen;
	oldpen = dc.SelectObject(&bdpen);
	dc.MoveTo(100, 100);
	dc.LineTo(500, 100);
	dc.MoveTo(500, 100);
	dc.LineTo(500, 500);
	dc.MoveTo(500, 500);
	dc.LineTo(100, 500);
	dc.MoveTo(100, 500);
	dc.LineTo(100, 100);
	dc.SelectObject(oldpen);

	dc.SelectStockObject(BLACK_PEN);
	dc.SelectStockObject(NULL_BRUSH);
	dc.Ellipse(100, 100, 300, 500);
```

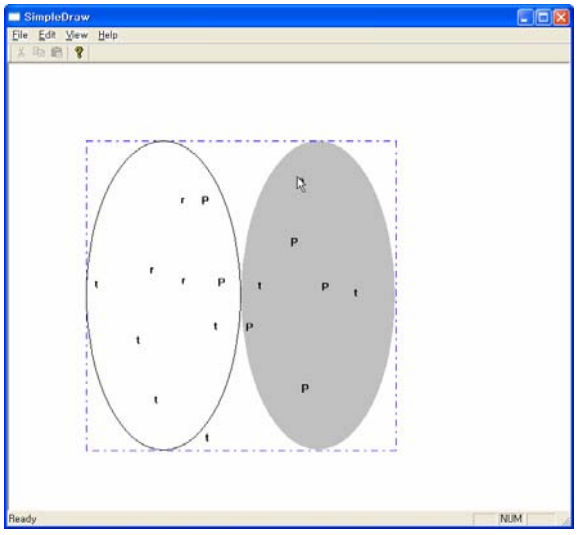
```
	dc.SelectStockObject(NULL_PEN);
	dc.SelectStockObject(LTGRAY_BRUSH);
	dc.Ellipse(300, 100, 500, 500);
	// Do not call CWnd::OnPaint() for painting messages
}

void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
	// TODO: Add your message handler code here and/or call default
	CClientDC dc(this); // device context for painting
	dc.SetBkMode(TRANSPARENT);
	dc.TextOut(point.x, point.y, m_str);
	CWnd::OnLButtonDown(nFlags, point);
}

void CChildView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	// TODO: Add your message handler code here and/or call default
	m_str.Format("%c", nChar);
	CWnd::OnChar(nChar, nRepCnt, nFlags);
}
```

12. 다음 간단한 Person 클래스의 예제에서 person2, person3, person4는 각각 Shallow Copy

화면에 파란색 대쉬도트모양의 선으로 된 사각형과, 검정색 선으로 된 타원과, 선이 없고 회색바탕으로 된 타원이 그려져 있으며, 키보드로 선택한 글자로 왼쪽 마우스를 클릭하면 마우스를 클릭한 위치에 글자가 출력되어 나타난다.



학과 _____ 학번 _____ 이름 _____

에 인한 메모리 참조 에러가 발생한다. 이 문제를 해결하는 방법을 적어라. (10점)
(힌트: DEEP COPY를 수행하는 copy constructor와 operator=을 작성한다)

```
#include <iostream>
using namespace std;

class Person {
	friend ostream& operator<<(ostream&, const Person&);
public:
Person();
	Person(char*, int);
	~Person();
	void Show();
private:
	Char *name;
	int age;
};

Person::Person()
{
	name = new char[10];
	strcpy(name, " ");
	age = 0;
}

Person::Person(char * n, int a)
{
	name = new char[strlen(n)+1];
	strcpy(name, n);
	age = a;
}

Person::~Person()
{
	delete [] name;
}

void Person::Show()
{
	cout << "Name: " << name << " Age: " << age << endl;
}

ostream& operator<<(ostream& out, const Person& p)
{
	out << "Name: " << p.name << " Age: " << p.age << endl;
	return out;
}

int main()
{
	Person person1("Kim", 20);
	cout << "Person1 : ";
	person1.Show();

	Person person2(person1);   // shallow copy 에 의한 메모리 참조에러발생
	Person person3 = person1;  // shallow copy 에 의한 메모리 참조에러발생

	Person person4;
	person4 = person1;         // shallow copy 에 의한 메모리 참조에러발생
	cout << "Person4 : " << person4;

	return 0;
}
```

학과 _____ 학번 _____ 이름 _____

```
class Person {
	Person(const Person&);		// DEEP COPY copy constructor
	Person& operator=(const Person&); // DEEP COPY operator=
};
Person::Person(const Person& p)
{
	name = new char[strlen(p.name)+1];
	strcpy(name, p.name);
	age = p.age;
}

Person& Person::operator=(const Person& p)
{
	if (this == &p) return *this;

	delete [] name;
	name = new char[strlen(p.name)+1];
	strcpy(name, p.name);

	age = p.age;

	return *this;
}
```

-끝-