**1. “WinMain” 能形成为“winmain” 吗？**

**答案**：不能。  
**解释**：在 Windows API (WinAPI) 中，WinMain 是程序的入口点，系统会寻找这个特定名称的函数。Windows API 对函数名是区分大小写的，因此必须使用大写 W 和大写 M 写作 WinMain。如果写成 winmain，系统将无法识别，导致程序无法启动。

**2. 窗口类要用 WndProc 能写成“wndproc”或其他符合函数命名的名字吗？**

**答案**：可以  
**解释**：窗口类的回调函数（窗口过程）的名称可以是任何合法的标识符，不一定非要叫 WndProc 或 wndproc。关键在于注册窗口类时，将正确的函数地址赋值给 WNDCLASS 结构的 lpfnWndProc 成员。窗口名称（lpszClassName）与回调函数的命名也没有关联，因此不会产生影响。

**3. “LRESULT CALLBACK WndProc” 函数中的 “break; 窗口子名出现时多个类是什么？**

**答案**：break 语句用于跳出 switch 语句块，确保在处理完某个消息后不再执行其他 case。对于多个窗口类，每个窗口类可以有自己的 WndProc 函数来处理消息。  
**解释**：在 WndProc 函数中，通常使用 switch(message) 来处理不同的消息。break 语句的作用是处理完一个 case 后跳出 switch，避免执行其他分支。

**4. “LRESULT CALLBACK WndProc” 函数中的 “default: return DefWindowProc(hwnd, message, wParam, lParam);” “缺少后会出现什么样的结果？**

**答案**：如果缺失 default: return DefWindowProc(hwnd, message, wParam, lParam);，未处理的消息将返回 0，这可能导致窗口行为异常。  
**解释**：DefWindowProc 是 Windows 提供的默认消息处理函数，用于处理程序未显式处理的消息。如果在 WndProc 的 switch 语句中没有 default 分支调用

**5. 怎样理解 “WinMain” 和窗口信息 “WINAPI”？**

**答案**：WinMain 是 Windows 程序的入口点，类似于 C++ 中的 main 函数；WINAPI 是一个宏，定义了函数的调用约定，通常是 \_\_stdcall。  
**解释**：

* WinMain：它是程序启动时系统调用的第一个函数，负责初始化窗口、注册窗口类并运行消息循环。它的典型签名是 int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)。
* WINAPI：这是一个宏，用于指定函数的调用约定为 \_\_stdcall，确保参数从右到左压栈，并由被调用函数清理堆栈。这是 Windows API 函数的标准约定。

**6. 这么理解 “LRESULT CALLBACK WndProc( HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam )” 中的 CALLBACK ？**

**答案**：CALLBACK 是一个宏，通常定义为 \_\_stdcall，用于指定回调函数的调用约定。  
**解释**：在 WndProc 函数的声明中，CALLBACK 表示该函数将以 \_\_stdcall 调用约定被 Windows 系统调用。这种约定规定了参数传递和栈清理的方式，确保 WndProc 能正确接收和处理系统发送的消息（如窗口创建、销毁等）。

**7. break; 改为 return 0; 有效有影响吗？为什么？**

**答案**：有影响。改为 return 0; 后，处理完该消息会直接返回，不会执行到 default: return DefWindowProc(...)，可能导致某些消息未被正确处理。  
**解释**：以下是您提供的代码片段的分析：

c

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch(message)

{

case ...:

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hwnd, message, wParam, lParam);

}

return 0;

}

* **正常情况**：break 跳出 switch 块后，函数继续执行到 return 0;，但未处理的消息会由 default 分支的 DefWindowProc 处理。
* **改为** return 0;：如果将 break 改为 return 0;，处理完某个 case（如 WM\_DESTROY）后会直接返回，跳过 default 分支。这意味着未处理的消息不会被 DefWindowProc 处理，可能导致窗口行为异常。例如，某些系统消息需要特定的返回值，而直接返回 0 可能不符合期望。

**8. “int WINAPI WinMain (HINSTANCE hThisInst, HINSTANCE hPrevInst, LPSTR lpszCmdLine, int nCmdShow)” 中用的 4 个参数的值是什么条件的？**

**答案**：这四个参数提供了程序运行时的环境信息：

* hThisInst：当前应用程序实例的句柄。
* hPrevInst：上一个应用程序实例的句柄，在 Win32 中通常为 NULL。
* lpszCmdLine：命令行参数字符串。
* nCmdShow：指定窗口的初始显示方式。  
  **解释**：
* hThisInst：由系统提供，用于标识当前运行的程序实例，常用于加载资源。
* hPrevInst：在 16 位 Windows 中用于区分多个实例，但在 32 位及以上系统中已无意义，始终为 NULL。
* lpszCmdLine：从命令行传递给程序的字符串，例如运行程序时的附加参数。
* nCmdShow：由系统或调用者指定，控制窗口显示状态（如 SW\_SHOW 表示正常显示，SW\_MAXIMIZE 表示最大化）。

**9. 为什么 WinMain 函数是怎样的？**

**答案**：WinMain 的结构是为了与 Windows 系统兼容，接收必要的参数来初始化程序并启动消息循环。  
**解释**：Windows 系统将 WinMain 定义为程序的入口点，而不是传统的 main，以适应图形用户界面 (GUI) 程序的需求。它需要接收实例句柄（hThisInst）、命令行参数（lpszCmdLine）和显示状态（nCmdShow）等信息，以便程序创建窗口、处理用户输入并运行消息循环。这是 Windows 编程的基本约定。

**10. 至少有课本知识的程序后运行，点击窗口右上角的 “X” 时，窗口产生结束要结束要按资源**

**答案**：在处理 WM\_CLOSE 消息时，应释放窗口相关的资源（如动态内存、GDI 对象等），然后调用 PostQuitMessage(0) 结束程序。  
**解释**：当用户点击窗口右上角的 “X” 按钮时，系统会发送 WM\_CLOSE 消息到 WndProc 函数。程序需要在该消息的处理中：

1. 释放分配的资源，例如动态内存（通过 free 或 delete）、文件句柄或 GDI 对象（如画刷、位图等）。
2. 调用 PostQuitMessage(0)，向消息队列发送 WM\_QUIT 消息，结束消息循环，从而关闭程序。  
   示例代码如下：

c

case WM\_CLOSE:

DestroyWindow(hwnd); // 销毁窗口

break;

case WM\_DESTROY:

PostQuitMessage(0); // 结束消息循环

break;