Practical No.2

Write a program to create window with menu bar.

a. Try to display a window with different combinations of window styles and observe the results.

Code:

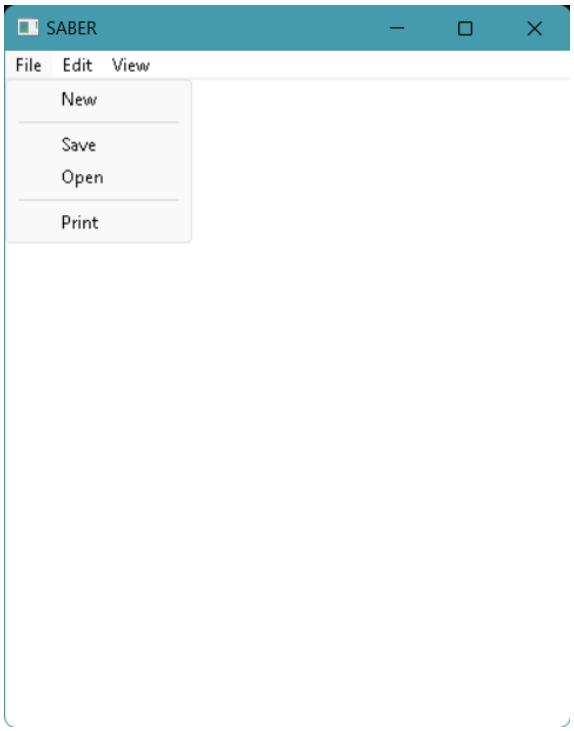
```
#include<windows.h>
#include<stdio.h>

void addMenu();
HWND hMenu;

LRESULT CALLBACK WindowProcedure(HWND hwnd, UINT uMsg, WPARAM
wparam, LPARAM lparam){
    switch(uMsg){
        case WM_QUIT:
            PostQuitMessage(0);
            break;

        case WM_CREATE:
            addMenu(hwnd);
            break;

        default:
            return DefWindowProc(hwnd, uMsg, wparam, lparam);
    }
}

void addMenu(HWND hwnd){
    hMenu = CreateMenu();
    HWND filemenu = CreateMenu();
    HWND editmenu = CreateMenu();
    HWND viewmenu = CreateMenu();
    HWND fontmenu = CreateMenu();

    AppendMenu(hMenu, MF_POPUP, (UINT_PTR)filemenu, "File");
    AppendMenu(hMenu, MF_POPUP, (UINT_PTR)editmenu, "Edit");
    AppendMenu(hMenu, MF_POPUP, (UINT_PTR)viewmenu, "View");
```

```
    AppendMenu(filemenu, MF_STRING, NULL, "New");
    AppendMenu(filemenu, MF_SEPARATOR, NULL, "");
    AppendMenu(filemenu, MF_STRING, NULL, "Save");
    AppendMenu(filemenu, MF_STRING, NULL, "Open");
    AppendMenu(filemenu, MF_SEPARATOR, NULL, "");
    AppendMenu(filemenu, MF_STRING, NULL, "Print");

    AppendMenu(editmenu, MF_STRING, NULL, "Undo");
    AppendMenu(editmenu, MF_STRING, NULL, "Redo");
    AppendMenu(editmenu, MF_SEPARATOR, NULL, "");
    AppendMenu(editmenu, MF_POPUP, (UINT_PTR)fontmenu, "Font
Style");

    AppendMenu(viewmenu, MF_STRING, NULL, "Toolbar");
    AppendMenu(viewmenu, MF_STRING, NULL, "Manager");
    AppendMenu(viewmenu, MF_STRING, NULL, "Word Wrap");

    AppendMenu(fontmenu, MF_STRING, NULL, "Bold");
    AppendMenu(fontmenu, MF_STRING, NULL, "Italic");
    AppendMenu(fontmenu, MF_STRING, NULL, "Underline");

    SetMenu(hwnd, hMenu);
}

int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE preinstance, LPSTR
lpstr, int cmdshow){

    const wchar_t CLASS_NAME[] = L"Sample Window Class";

    WNDCLASS wc = {};
    wc.lpszClassName = CLASS_NAME;
    wc.lpfnWndProc = WindowProcedure;
    wc.hInstance = hinstance;

    RegisterClass(&wc);

    HWND hld;
    hld = CreateWindow(CLASS_NAME, "SABER", WS_OVERLAPPEDWINDOW,
100, 100, 400, 500, 0, 0, hinstance, 0);
    ShowWindow(hld, cmdshow);

    MSG msg = {};
```

```
    while(GetMessage(&msg, NULL, 0, 0) > 0){
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return 0;
}
```

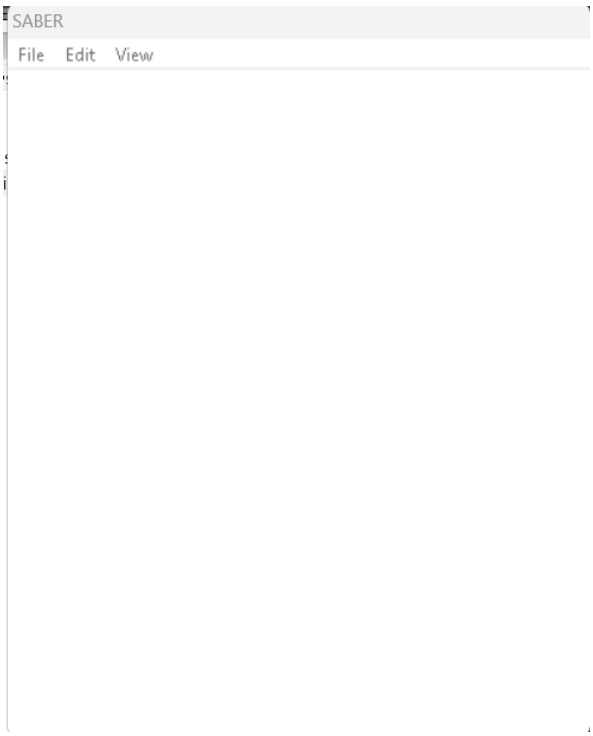
Output:

WS_OVERLAPPEDWINDOW



WS_CAPTION

WS_DISABLED



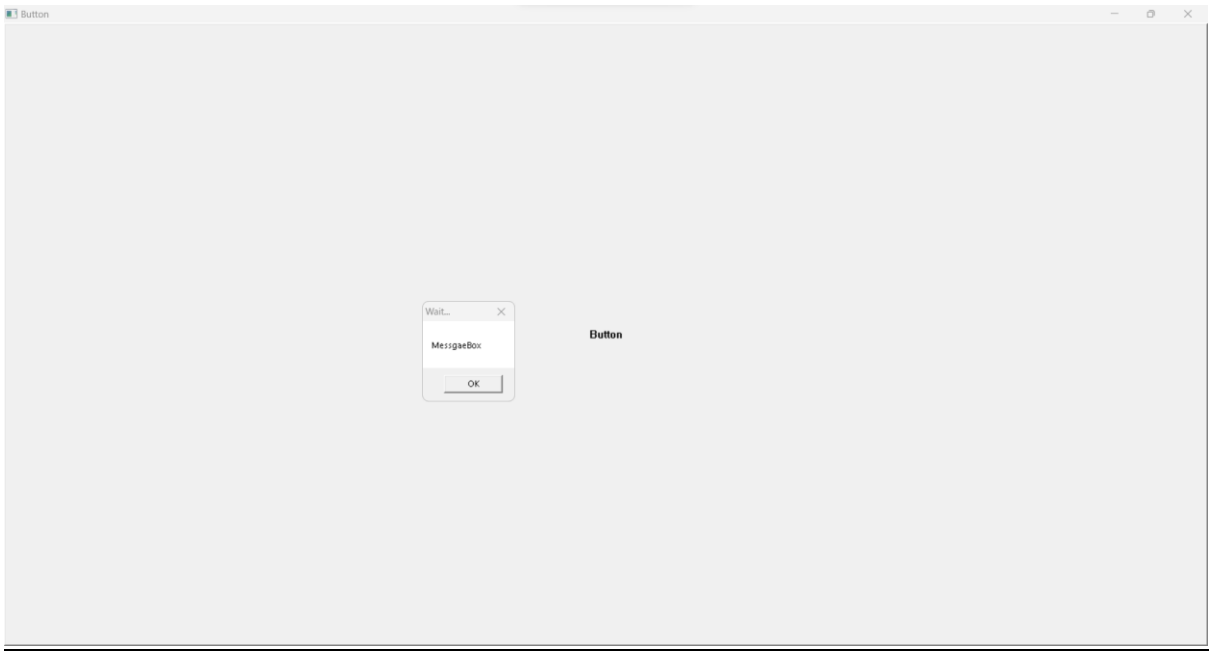
b. Write a program, which receives an integer as a command line argument, creates edit window, and based on the value of the integer displays button window as maximized / minimized / normal.

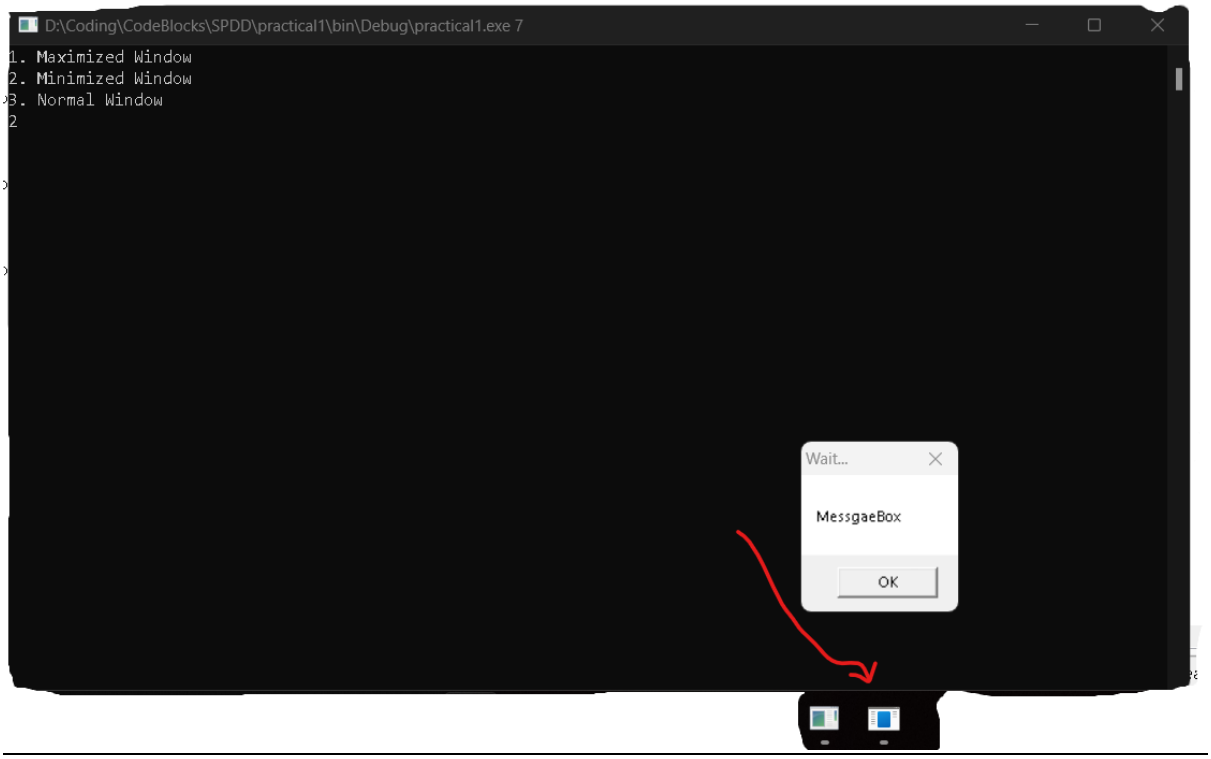
Code:

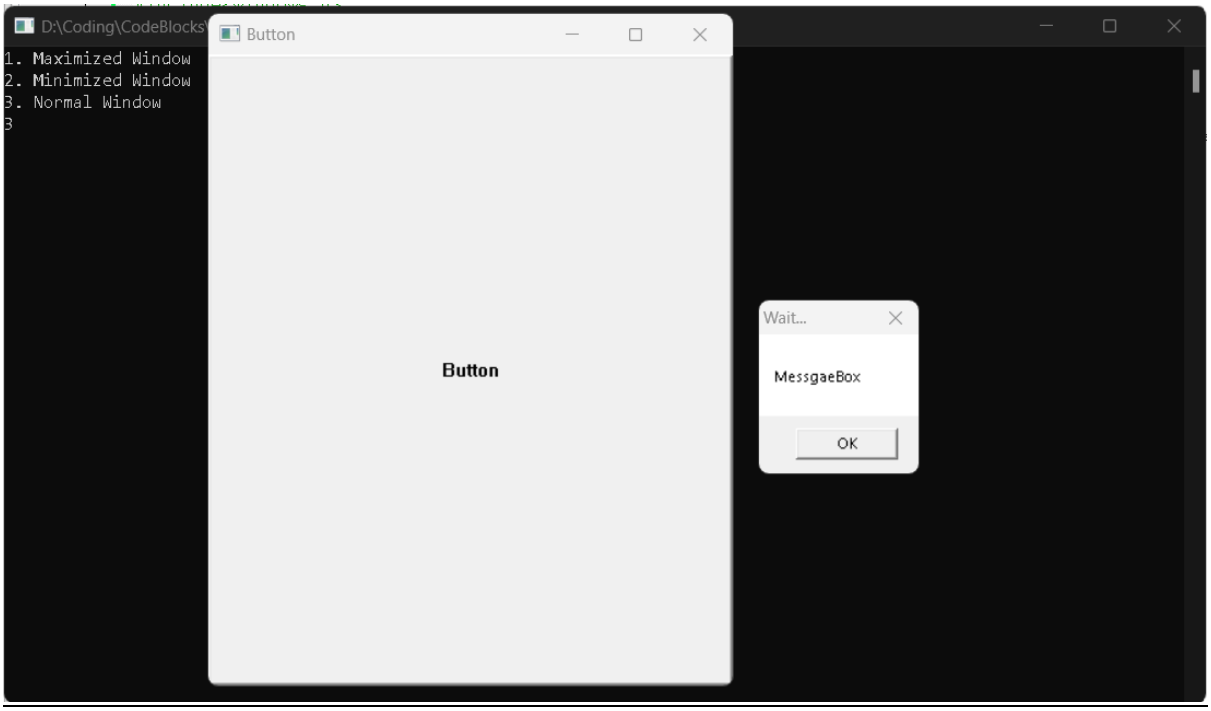
```
#include<windows.h>
#include<stdio.h>

int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE preinstance, LPSTR
lpstr, int cmdshow){
    HWND hld;
    int ch;
    hld = CreateWindow("Button", "Button", WS_OVERLAPPEDWINDOW, 100,
100, 400, 500, 0, 0, hinstance, 0);

    printf("1. Maximized Window\n");
    printf("2. Minimized Window\n");
    printf("3. Normal Window\n");
    scanf("%d", &ch);
    switch(ch){
        case 1:
```

```
            ShowWindow(hld, 3);
            break;

        case 2:
            ShowWindow(hld, 2);
            break;

        case 3:
            ShowWindow(hld, 1);
            break;

        default:
            printf("Invalid Choice!");
    }

    MessageBox(0, "MessgaeBox", "Wait...", MB_OK);

    return 0;
}
```

Output:




D:\Coding\CodeBlocks\SPDD\practical1\bin\Debug\practical1.exe 7
1. Maximized Window
2. Minimized Window
3. Normal Window
2
Wait...
MessgaeBox
OK


D:\Coding\CodeBlocks
1. Maximized Window
2. Minimized Window
3. Normal Window
3
Button
Button
Wait...
MessgaeBox
OK