```cpp
#define UNICODE
#include<windows.h>
#include"ClassFactoryDllServerWithRegFile.h"
// class declarations
class CSumSubtract:public ISum,ISubtract
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CSumSubtract(void);
    // destructor method declarations
    ~CSumSubtract(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // ISum specific method declarations (inherited)
    HRESULT __stdcall SumOfTwoIntegers(int,int,int *);
    // ISubtract specific method declarations (inherited)
    HRESULT __stdcall SubtractionOfTwoIntegers(int,int,int *);
};
class CSumSubtractClassFactory:public IClassFactory
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CSumSubtractClassFactory(void);
    // destructor method declarations
    ~CSumSubtractClassFactory(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // IClassFactory specific method declarations (inherited)
    HRESULT __stdcall CreateInstance(IUnknown *,REFIID,void **);
    HRESULT __stdcall LockServer(BOOL);
};
// global variable declarations
long glNumberOfActiveComponents=0;// number of active components
long glNumberOfServerLocks=0;// number of locks on this dll
// DllMain
BOOL WINAPI DllMain(HINSTANCE hDll,DWORD dwReason,LPVOID Reserved)
{
    // code
    switch(dwReason)
    {
    case DLL_PROCESS_ATTACH:
        break;
    case DLL_PROCESS_DETACH:
        break;
```

```
53      }
54  return(TRUE);
55  }
56  // Implementation Of CSumSubtract's Constructor Method
57  CSumSubtract::CSumSubtract(void)
58  {
59      // code
60      m_cRef=1;// hardcoded initialization to anticipate possible failure of
          QueryInterface()
61      InterlockedIncrement(&glNumberOfActiveComponents);// increment global counter
62  }
63  // Implementation Of CSumSubtract's Destructor Method
64  CSumSubtract::~CSumSubtract(void)
65  {
66      // code
67      InterlockedDecrement(&glNumberOfActiveComponents);// decrement global counter
68  }
69  // Implementation Of CSumSubtract's IUnknown's Methods
70  HRESULT CSumSubtract::QueryInterface(REFIID riid,void **ppv)
71  {
72      // code
73      if(riid==IID_IUnknown)
74          *ppv=static_cast<ISum *>(this);
75      else if(riid==IID_ISum)
76          *ppv=static_cast<ISum *>(this);
77      else if(riid==IID_ISubtract)
78          *ppv=static_cast<ISubtract *>(this);
79      else
80      {
81          *ppv=NULL;
82          return(E_NOINTERFACE);
83      }
84      reinterpret_cast<IUnknown *>(*ppv)->AddRef();
85      return(S_OK);
86  }
87  ULONG CSumSubtract::AddRef(void)
88  {
89      // code
90      InterlockedIncrement(&m_cRef);
91      return(m_cRef);
92  }
93  ULONG CSumSubtract::Release(void)
94  {
95      // code
96      InterlockedDecrement(&m_cRef);
97      if(m_cRef==0)
98      {
99          delete(this);
100         return(0);
101     }
102     return(m_cRef);
103 }
```

```
104 // Implementation Of ISum's Methods
105 HRESULT CSumSubtract::SumOfTwoIntegers(int num1,int num2,int *pSum)
106 {
107     // code
108     *pSum=num1+num2;
109     return(S_OK);
110 }
111 // Implementation Of ISubtract's Methods
112 HRESULT CSumSubtract::SubtractionOfTwoIntegers(int num1,int num2,int *pSubtract)
113 {
114     // code
115     *pSubtract=num1-num2;
116     return(S_OK);
117 }
118 // Implementation Of CSumSubtractClassFactory's Constructor Method
119 CSumSubtractClassFactory::CSumSubtractClassFactory(void)
120 {
121     // code
122     m_cRef=1;// hardcoded initialization to anticipate possible failure of
      QueryInterface()
123 }
124 // Implementation Of CSumSubtractClassFactory's Destructor Method
125 CSumSubtractClassFactory::~CSumSubtractClassFactory(void)
126 {
127     // code
128 }
129 // Implementation Of CSumSubtractClassFactory's IClassFactory's IUnknown's
    Methods
130 HRESULT CSumSubtractClassFactory::QueryInterface(REFIID riid,void **ppv)
131 {
132     // code
133     if(riid==IID_IUnknown)
134         *ppv=static_cast<IClassFactory *>(this);
135     else if(riid==IID_IClassFactory)
136         *ppv=static_cast<IClassFactory *>(this);
137     else
138     {
139         *ppv=NULL;
140         return(E_NOINTERFACE);
141     }
142     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
143     return(S_OK);
144 }
145 ULONG CSumSubtractClassFactory::AddRef(void)
146 {
147     // code
148     InterlockedIncrement(&m_cRef);
149     return(m_cRef);
150 }
151 ULONG CSumSubtractClassFactory::Release(void)
152 {
153     // code
```

```
154     InterlockedDecrement(&m_cRef);
155     if(m_cRef==0)
156     {
157         delete(this);
158         return(0);
159     }
160     return(m_cRef);
161 }
162 // Implementation Of CSumSubtractClassFactory's IClassFactory's Methods
163 HRESULT CSumSubtractClassFactory::CreateInstance(IUnknown *pUnkOuter,REFIID
      riid,void **ppv)
164 {
165     // variable declarations
166     CSumSubtract *pCSumSubtract=NULL;
167     HRESULT hr;
168     // code
169     if(pUnkOuter!=NULL)
170         return(CLASS_E_NOAGGREGATION);
171     // create the instance of component i.e. of CSumSubtract class
172     pCSumSubtract=new CSumSubtract;
173     if(pCSumSubtract==NULL)
174         return(E_OUTOFMEMORY);
175     // get the requested interface
176     hr=pCSumSubtract->QueryInterface(riid,ppv);
177     pCSumSubtract->Release();// anticipate possible failure of QueryInterface()
178     return(hr);
179 }
180 HRESULT CSumSubtractClassFactory::LockServer(BOOL fLock)
181 {
182     // code
183     if(fLock)
184         InterlockedIncrement(&glNumberOfServerLocks);
185     else
186         InterlockedDecrement(&glNumberOfServerLocks);
187     return(S_OK);
188 }
189 // Implementation Of Exported Functions From This Dll
190 HRESULT __stdcall DllGetClassObject(REFCLSID rclsid,REFIID riid,void **ppv)
191 {
192     // variable declaraions
193     CSumSubtractClassFactory *pCSumSubtractClassFactory=NULL;
194     HRESULT hr;
195     // code
196     if(rclsid!=CLSID_SumSubtract)
197         return(CLASS_E_CLASSNOTAVAILABLE);
198     // create class factory
199     pCSumSubtractClassFactory=new CSumSubtractClassFactory;
200     if(pCSumSubtractClassFactory==NULL)
201         return(E_OUTOFMEMORY);
202     hr=pCSumSubtractClassFactory->QueryInterface(riid,ppv);
203     pCSumSubtractClassFactory->Release();// anticipate possible failure of
        QueryInterface()
```

```
204     return(hr);
205 }
206 HRESULT __stdcall DllCanUnloadNow(void)
207 {
208     // code
209     if((glNumberOfActiveComponents==0) && (glNumberOfServerLocks==0))
210         return(S_OK);
211     else
212         return(S_FALSE);
213 }
214
```