ClassName.hpp

```
// Contains class template definition

template<U, V>
class ClassName(){
  U member1(V){
  ...
  }
  ...

}
```

ClassName_type1_type2.hpp

```
// Contains extern specialisation declarations
// which should be included when class is
// used in dependent code

#include <ClassName.hpp>

// where type1,type2 are types defined at the
// time of library compilation
// usually they would be fundamental types
extern template class ClassName<type1, type2>;

extern template
type1 ClassName<type1, type2>::member1(type2);
```

ClassName_type1_type2.cpp

```
// Contains specialisation declarations
// used to build .so

#include <ClassName.hpp>

// where type1,type2 are types defined at the
// time of library compilation
// usually they would be fundamental types
template class ClassName<type1, type2>;

template
type1 ClassName<type1, type2>::member1(type2);
```

Compile

main.cpp

```
// File used by library user to
// to build own binary

// type1/type2 would be fundamental types
#include <ClassName_type1_type2.hpp>

int main(){
  typedef ClassName_type1_type2 myClass;

  myClass test;
  type2 a=5;

  test.member1(a);

  return 0;
}
```

Lib.so

Link