//template

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

template<class T>
void mSwap(T& a,T& b){

   T temp = a;
   a = b;
   b = temp;

}


int main () {

   int a = 1, b = 2;
   mSwap<int>(a,b);
   cout<<a<<" "<<b<<endl;
   char aa = 'a';
   char bb = 'b';
   mSwap<char>(aa,bb);
   cout<<aa<<" "<<bb<<endl;

}
```

//glimpse of problems when using row pointers

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

class TestObj{
public:
   int x;
   void print(){
      cout<<"a print func\n"<<x<<endl;
   }
   TestObj(int i): x(i){}
   ~TestObj(){
      cout<<"destructor invoked\n";
```

```
    }
};

int main(){

    TestObj* a = new TestObj(1);
    TestObj* b = a;
    delete a;
    TestObj* c = new TestObj(2);
    b->print();
}
```

//shared_ptr

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

class TestObj{

public:
    TestObj(){}
    ~TestObj(){
        cout<<"destructor invoked\n";
    }

};

int main(){


    shared_ptr<TestObj> b;
    {
        shared_ptr<TestObj> a = make_shared<TestObj>();
        b = a;
        cout<<b.use_count()<<" pointer/s pointing to the object\n";
    }
    cout<<"I am here, ptr \"a\" is terminated, does the destructor invoked above ? \n";
    cout<<b.use_count()<<" pointer/s pointing to the object\n";

}
```

// unique_ptr

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

class TestObj{

public:
   TestObj(){}
   ~TestObj(){
      cout<<"destructor invoked\n";
   }

};

int main(){


   unique_ptr<TestObj> b;
   {
      unique_ptr<TestObj> a = make_unique<TestObj>();
      // we can not do this
      //b = a;
   }
   cout<<"I am here, ptr \"a\" is terminated, does the destructor invoked above ? \n";

}
```

// functor

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

class func{
public:
   //state
   int a;
   int operator()(int i){
      return a + i;
```

```
    }
};

int main(){

    func functor;
    functor.a = 10;
    int res = functor(1);
    cout<<res<<endl;
    res = functor(2);
    cout<<res<<endl;
}
```

//function pointers, lambda functions

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


void print(){
    cout<<"hello, world\n";
}


int main(){

    void(*ptr)() = print;
    ptr();

    int(*ptr2)(int, int) = [](int a, int b) -> int { return a + b;};
    cout<<ptr2(1,7)<<endl;
}
```

```
//An example
//function pointers, passing a function,
//lambda functions
#include <iostream>
#include <vector>
#include <memory>
using namespace std;

void mSort(vector<int>& v, bool(*f)(int ,int)){
   for(int i = 0; i < v.size(); i++){
      int temp = i;
      for(int j = i + 1; j < v.size(); j++){
         if(f(v[j], v[temp]))
            temp = j;
      }
      swap(v[temp], v[i]);
   }
}

int main(){
   vector<int> vec(10);
   srand(1);
   for(int i = 0; i < 10; i++)
      vec[i] = rand()%1000;
   mSort(vec, [](int a,int b){return a > b;});
   for(int i = 0; i < vec.size();i++)
      cout<<vec[i]<<" ";
}
```