1. Vector is a sequential container to store elements. Vectors can grow dynamically whereas arrays have fixed sizes.
Vector takes more memory and takes longer time to access elements than an array.Array access is always constant time because the elements are arranged in a contigous memory location.

```
#include <vector>
int main{
        std::vector<int> yum;
        yum.push_back(458);
        yum.pop_back();
}
```

2. Iterator is any object that, pointing to some element in a range of elements, has the ability to iterate through the elements of that range.On the other hand, a pointer is a variable which stores the address of another variable.We can perform simple arithmetic operations on pointers but iterators are more restricted. Pointers can be dymanically alloted and deleted using delete but there is no concept of delete for iterators.

```
std::vector<std::pair<int,int>> :: iterator it;

for(it=pair.begin();it!=pair.end();it++){
        std::cout<<it->first<<" "<<it->second<<"\n";
```

3. The std::pair is a simple container consisting of two elements.The first element is referenced as first and the second as second.Pair can store two heterogeneous elements as single entity.
A good replacement for pair will be a structure with two elemnts.

```
struct Pair{
        int first;
        int second;
};
```

4. Maps are associated containers that store elements in a mapped fashion following a specific order. Each element has a key value and a mapped value.All the key values have to be unique. The elements are sorted by its keys indicated by an internal compare object, by default it sorts in an ascending order.

```
#include <map>
int main(){
std::map<int, int> yum;
yum.insert(std::pair<int,int> (1,234);
yum.insert(std::pair<int,int> (2,2);
yum.insert(std::pair<int,int> (3,35);
yum.insert(std::pair<int,int> (4,24);

for(auto it: yum){
        std::cout << '\t' << itr->first
```

```
            << '\t' << itr->second << '\n';
}
}
```

5. Sets are associated containers which each element of the container will be unique just like the mathematical sets. The value of the set identifies itself. The value of the set cannot be modified once it is added to the set but it is possible toremove or add the modified value of the element.
The values are stored in a sorted order just like std::set and is indicated by an internal compare object and by default it sorts in an ascending order.

```
#include <set>
int main(){
        std::set<int> s;
        s.insert(3);
        s.insert(6);
        s.insert(4);
        s.insert(78);
        s.insert(10);
        for(auto it:s)
                std::cout<<it<<" ";
}
```