```cpp
//////// Lecture 92: Dynamic Arrays (std::vector)
#include <iostream>
// suppose we need to store a list of numbers in an array.
int main(){
    int arr[10];
}
// We can hold no more than 10 integers. What if, at runtime, we can store more than 10 integers?
// then we cannot use this array, because it cannot automatically grow.
// In this case, we'll have to create a dynamic array and we'll have to allocate the memory ourselves.

int main(){
    int* ptr = new int[10];
    for (int i = 0;i < 10; ++i){
        ptr[i] = i*10; // some data
    }
}
/*What if at runtime, we need to store more than 10 integers in the array? We will have to
  1)allocate new memory that is large enough to accomodate all the old elements and the new elements,
  2) and then we need to copy the elements from the old array into the new array,
  3) and free the memory of the old array.

// Since this involves manual memory management, it's easy to make mistakes. If you need a behaviour of dynamic array, you should use the vector class.
*/
#include <vector>
// <> are called angular brackets.
std::vector<TYPE> varname
// you may use the initialisation_list syntax to declare a vectors using numbers that you already have at compile time.
std::vector<int> varname{1,2,3};

/// General methods you should know:
```

```cpp

// STL VECTORS: Iterators
// To create an iterator, we use auto and then we call the begin() function.
auto it = vec.begin();
// Creates an iterator initialises to the beginning of the array, and then returns the object.
// To access an element:
int object = *it;
// You may also modify the element in place:
*it = 2; // overwritten.
// Increment operator and decrement operator
++it; // shifts it to the next element.
--it; // shifts it to the previous element.
// It also overloads the + operator, so we can jump to any position we want in the container.
it = it + 2;
// Erasing an element
vec.erase(it); // removes the first element.
// Inserting an element
vec.insert(/*position in the form of a vector*/,/*value of element to insert*/);
// for example:
vec.insert(vec.begin()+5, 500); // jump to the fifth location and insert the value 500.

```