	STL Vector/ Vec<T>	STL List/ dslist<T>	Singly-LL	Doubly-LL
O(n)	Size: O(1) push_back: O(1) erase: O(n) insert: O(n) pop_back: O(1) resize: O(n) operator=: O(n) clear: O(n) sort: O(nlog(n))	Size: O(1) push_back/push_front: O(1) erase: O(1) insert: O(1) pop_back/pop_front: O(1) resize: O(n) operator=: O(n) clear: O(n) sort: O(nlog(n))	Size: O(n) push_back: O(n) erase: O(1) insert: O(1) pop_back: O(n)	Size: push_back: erase: insert: pop_back:
Initialize	std::vector<int> vname;	std::list<int> Lname	Node_class *head = NULL;	
Insert	vname.insert(itr_i, val)	Lname.insert(itr_i, val)	Insert B into T: B->next= T->next; T->next = B;	B->prev = T; T=t->next; B->prev->next = B; T->prev = B; T->prev->next = T;
	When iterating: returns iterator to same place in list, but really the next element: ➔ name.insert(itr–) ➔ name.erase(itr--)			
erase	Vname.erase(itr_position) Erase can invalidate the iterator.	Lname.erase(itr_position) Use instead of vector (if a lot)	P = T; T = T->next P->next= T->next delete temp	P = T T = T->next; P->next = T->next P->next->prev=P delete T
Push_back	Sometimes alloc new array size 2*m_alloc	.Push_back(value) ←	P->next = T; T->next = NULL	P->next = T; P->next->prev = P T->next = NULL;
Pop_back	Vname.pop_back()	Lname.pop_back()	P = T T = T->next P->next = NULL	P = T T = T->next P->next->prev = NULL P->next = NULL
	Does not return value		Pop_back/push_back/Insert/Erase -> First iterate to the point you want to remove **-1**	
Iterators	Vector<int>::iterator v_itr = v.begin() Has [] Uses operator< vector<T>::insert() returns an itr b/c insert() may invalidate le itr passed in	list<int>::iterator l_itr = l.begin() No [] Uses operator< Incrementing the end() iterator in any STL list has undefined behavior	N/A	

Recursion:

- Infinite recursion can == segmentation fault

```
void fun(int n){
     if(n>0){
         print(n)
         fun(n-1);
     }
}
```

5 4 3 2 1

```
void fun(int n){
     if(n>0){
         fun(n-1);
         print(n);
     }
}
```

1 2 3 4 5

Sorting Time Complexities	
Binary	O(log(n))
Merge	O(nlog(n))
Insertion	O(n^2)
Bubble	O(n^2)
Recursive is the same	

Pointers:

- Using operator[] on a pointer is the same as using pointer arithmetic and then dereferencing the result.
- Writing int* x=5; will result a compiler error.
-

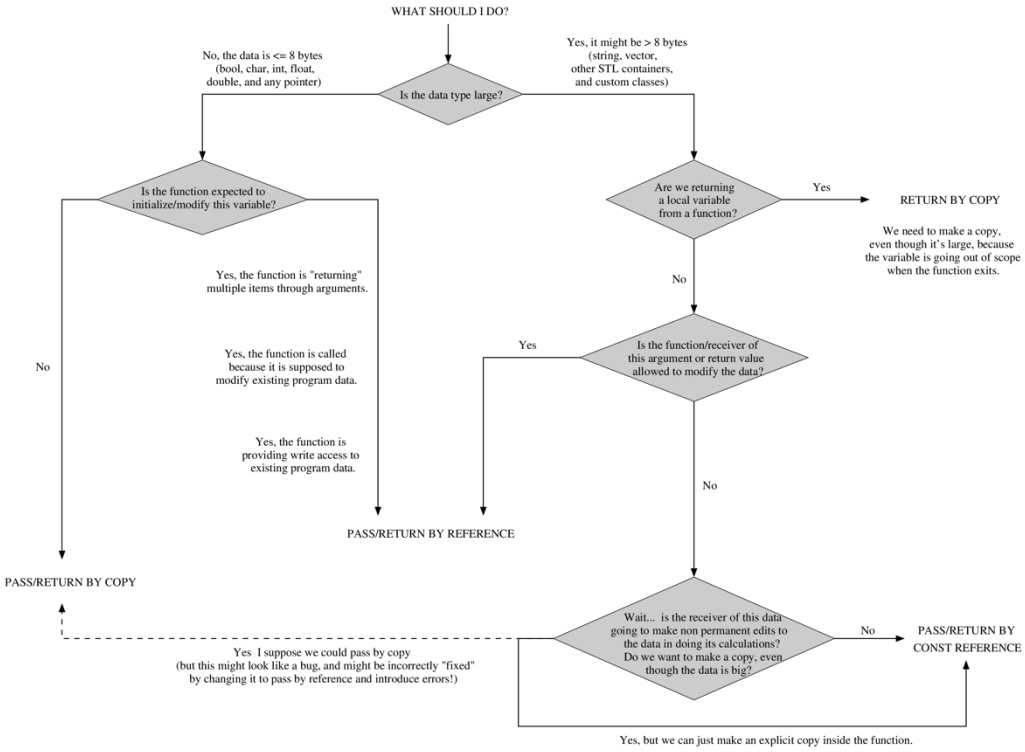
Classes:

Declaration → Class object_name(argument);

```
Class Date{
public:
  //Constructor
    Date();
    Date(int aMonth, int aDay, int aYear);
  //Accessor
    int getDay() const;
  //Modifier
    void setday(int bday);
  //Other member functions
    bool isEqual(const Date& *date2) const;
    int lastDayInMonth() const;
    void print() const;
Private:
    int day;
};
```

Implementation:

```
bool Date::isLeapYear() const{
}
```



- Use of uninitialized memory
- Reading/writing memory after it has been free'd (*NOTE: delete calls free*)
- Reading/writing off the end of malloc'd blocks (*NOTE: new calls malloc*)
- Reading/writing inappropriate areas on the stack
- Memory leaks - where pointers to malloc'd blocks are lost forever
- Mismatched use of malloc/new/new [] vs free/delete/delete []
- Overlapping src and dst pointers in memcpy() and related functions