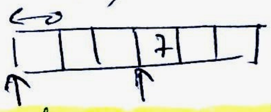
CS 107 lecture 5

generic with fcn pointer

```
void * lsearch (void *key, void * base, int n,
                int elemSize, int (* cmpfn)(void*, void*))
{
  for (int i=0; i<n; i++) {
      void * elemAddr = (char *) base + i * elemSize ;
      if (cmpfn (key, elemAddr) == 0)
          return elemAddr;
  }
  return NULL;
}
```

int example:

```
int array[] = {4, 2, 3, 7, 4, 6};
int size = 6;
int number = 7;
```

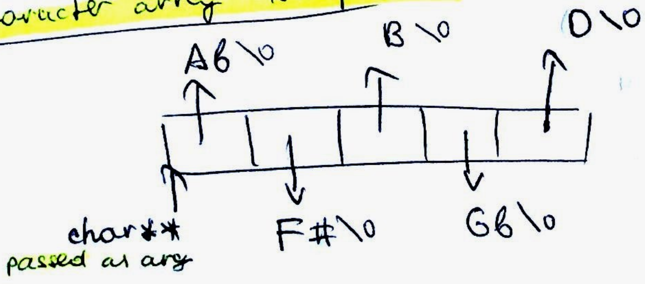


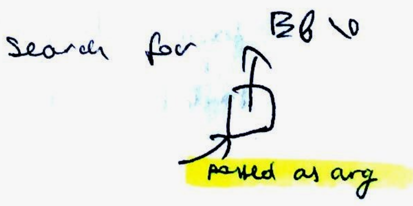
```
int* found = lsearch (&number, array, size, sizeof(int),
                      IntCmp);
```

```
int IntCmp (void * elem1, void * elem2)
{
    int * ip1 = elem1;
    int * ip2 = elem2;
    return *ip1 - *ip2;
}
```

} reinterpret void* as int* because we know in this case void* are int*

character array example:





```
char * notes [] = {"Ab", "F#", "B", "Gb", "D"};      array of pointers to strings
char * favoriteNote = "Eb"                            pointer to a string
                      pointer to pointer   pointer to array of pointers
char ** found = lsearch (&favoriteNote, notes, 5,
                         sizeof (char *), StrCmp);
int StrCmp (void * vp1, void * vp2)
{
    char * s1 = * (char **) vp1;
    char * s2 = * (char **) vp2;  [redacted]
                   recast to what vp1 and vp2 really are
    return strcmp (s1, s2);   [redacted]
}
```

Prototype for binary search (built-in)

```
void * bsearch (void * key, void * base, int n,
                int elemsize, int (* cmp)(void *, void *))
```

cannot have this pointer [redacted] implicitly passed
→ global fnxn or static method (no this pointer passed around)

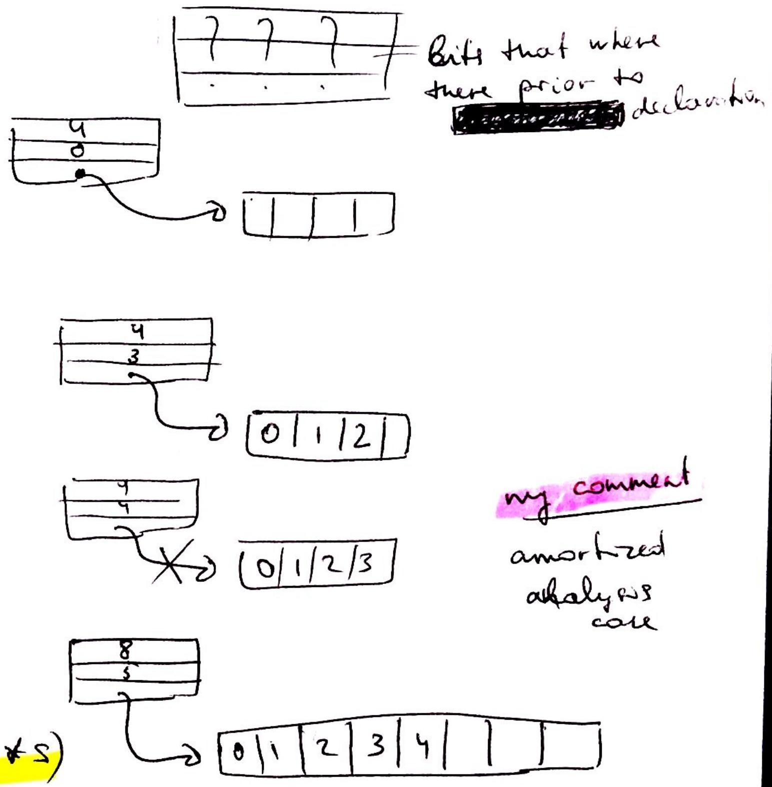
Generic data structure implementation

stack.h

```
typedef struct {
    int * elems;
    int logicalLen;
    int alloclength;
} stack;
void  StackNew (stack *s);
void  StackDispose (stack *s);
void  StackPush (stack *s, int value)
int   StackPop (stack *s)
```

```
stack s;
StackNew (&s);

for (int i=0; i<5; i++){
    StackPush (&s, i);
}
StackDispose (&s);
```



my comment
amortized analysis case

```
void StackNew (stack *s)
{
    s -> logicallen = 0;
    s -> alloclen = 4;
    s -> elems = malloc(4 * sizeof(int));
    assert (s -> elems != NULL);
}
```

malloc:
searches for a chunk of heap that has the # of bytes and returns its address

[redacted]
remove in production code

macro
if the test is false ends program and reports at what line the program ended
→ make sure malloc found space and did not return a NULL pointer