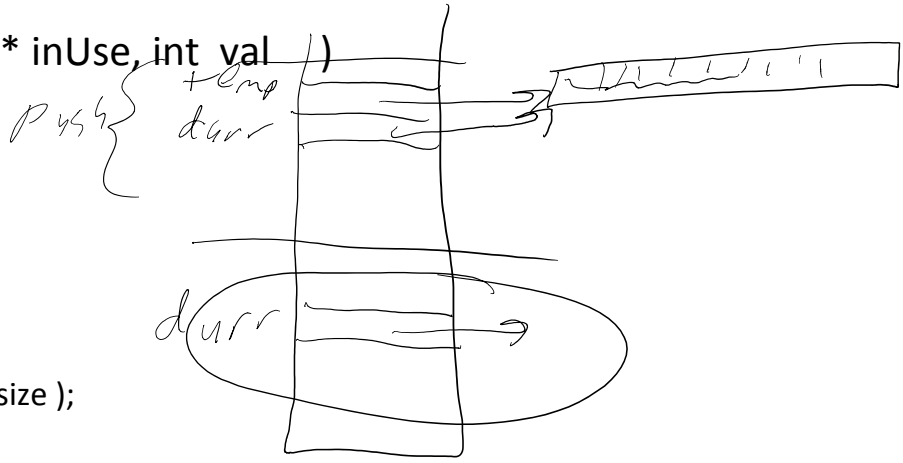
CS 211 - 09/12/17

Tuesday, September 12, 2017 4:46 PM

```
/* flawed version of push */
/*  darr is sent as pass-by-value*/
void push (   int *darr, int* size, int* inUse, int  val   )
{
  int localInUse = *inUse;

   /* check if array is full */
   if ( localInUse == *size)
   {
     int *temp;
     temp = (int *) malloc ( 2 * sizeof(int) * *size );
     int i;
     for ( i = 0 ; i < *size ; i++)
          temp[i] = darr[i];
     free (darr);
     darr = temp;
     *size = *size * 2;
/* From <
https://www3.cs.uic.edu/pub/CS211/ProjectF17/proj1f17.pdf
> */
      }
  darr[localInUse] = input;
  (localInUse)++;

  *inUse = localInUse;
}
```



From <https://www3.cs.uic.edu/pub/CS211/NotesF17/CS_211_-_0905.pdf>

```
main(   )
{
  int size = 10;
  int* darr = (int* ) malloc ( sizeof(int) * size);
  int inUse = 0;
  int val = 0;

  while ( val != -999)
   {
      scanf ("%d", &val);

      push ( darr, &size, &inUse, val);
   }
```

```
/* corrected version uses C style pass-by-reference for the array */
/*  the code to call the function */
    push2 (&darr, &size, &inUse, val);

/* corrected function code - used "double pointer" */
void push2 (   int **darr, int* size, int* inUse, int  val     )
{
  int localInUse = *inUse;
  int* localDArr = *darr;   /* to avoid dereference every time darr is used */

  /* check if array is full */
  if ( localInUse == *size)
  {
    int *temp;
    temp = (int *) malloc ( 2 * sizeof(int) * *size );
    int i;
    for ( i = 0 ; i < *size ; i++)
       temp[i] = localDArr[i];
    free (localDArr);
    localDAarr = temp;
    *size = *size * 2;
   }
 localADrr[localInUse] = input;
 (localInUse)++;

 *inUse = localInUse;
 *darr = localDArr;
}
```