Statistics 243: *class notes*

William J. De Meo

9/19/97

1 Memory Allocation

```
#inlcude <stdlib.h>

double *dmalloc(long n)
{
     double *x;

     if ((x=(double*) malloc((unsigned)(n*sizeof(double)))) == NULL)
     {
          printf("Error:  Couldn't get %ul bytes of memory\n",
          (unsigned)(n*sizeof(doulbe)));
          exit(1);
     }
return(x);
}
```

Save the function in a file called `memory.c` and compile it with:

```
cc -c memory.c
```

which results in a file called `memory.o`. Now we don't need to compile the memory allocation program again. When we have the listing:

```
double * dmalloc(long n); /* function declaration */
double * y;
y= dmalloc(n);
```

we can simply use

```
cc -o myprog myprog.c memory.o
```

Suppose we want a function which finds the maximum of an array of doubles. Could return the max through the function name, and return the index by modifying an input argument. Some people like to write functions which only return either 1 or 0 depending on if the function was successful or not. In that case, you would have to pass two address references to the function. We now describe the first model:

```
double getmax(double *x, long n, long *index)

long i, imax; /* i is the index counter */
      /* imax is the temporary max value */
```

```
double xmax;

imax = 0;
xmax = x[0]; /* This syntax is helpful (intead of *x) */
      /* Because it reminds us that we're using */
      /* a pointer to an array */
for(i=1;i<n;i++){
if(x[i]>xmax){
xmax=x[i];
imax=i;
}}

index = imax;
return(xmax);
}
```

So how would we call this function?

```
double x[1000];

double getmax(double *, long, long *); /* function prototype */
   /* note that, we don't need to actually */
   /* declare a pointer to long to be passed /*
   /* to getmax().  We can just declare a long */
   /* variable and pass it's address.  */

double themax;
long theindex;

themax = getmax(x, 1000, &theindex);
```

A better `dmalloc()` function would look like:

```
int dmalloc2(long n, double **x)
{
x = (double *) ...
}
```

Now we would call it with

```
double *x;
i = dmalloc1(n,&x);
```