Lecture 15: MPI Introduction + Assignment 2/Exam 1 solution review + class fireside chat

CMU 15-418: Parallel Computer Architecture and Programming (Spring 2012)

Assignment 4: WSP in MPI

- **In assignment 4 you will re-implement your WSP program using message passing, rather than shared address space, primitives**

- **The point is to think about how orchestrating the program by sending and receiving messages makes you think a bit differently than when orchestrating it with synchronized access to shared variables**

- **Programming models matter!**

Basic MPI

- **MPI: message passing interface**
  - API + runtime implementing message passing model
  - C and C++ variant available. Below I'm showing the C version of the API

```
$ mpirun -np 16 ./myprogram
```

run program from shell using `mpirun` (notice the number of processes to use is specified at command line)

16 instances of program are invoked by mpirun. Think SPMD!

```
#include <mpi.h>

void main(int argc, char** argv)  {

   MPI_Init(&argc, &argv);

   int numProcs;
   int procId;

   // get total number of processes
   MPI_Comm_size(MPI_COMM_WORLD, &numProcs);

   // get this processes' process id
   MPI_Comm_rank(MPI_COMM_WORLD, &procId);

   printf("I am process %d of %d\n", procId, numProcs);

   // each process must tell MPI it is quitting.  This is
   // required. Otherwise program will appear to hang at
   // the end (can think of MPI_Finalize as a required
   // global barrier at the end of every MPI program)
   MPI_Finalize();
}
```

Sending a message to another process

This is an example of blocking send and receive

```
#include <mpi.h>

void main(int argc, char** argv)  {

   MPI_Init(&argc, &argv);

   int numProcs;
   int procId;
   const int MY_MSG_TAG = 0;

   MPI_Comm_size(MPI_COMM_WORLD, &numProcs);
   MPI_Comm_rank(MPI_COMM_WORLD, &procId);

   int numElements = 1024;
   int* buf = new int[numElements];

   if (procId == 0) {
      // initialize buf ...

      // send message to process 1
      MPI_Send(buf, numElements, MPI_INT, 1, MY_MSG_TAG, MPI_COMM_WORLD);
   } else {
      // recv message.  Will match message from any source, as long as it has correct tag
      MPI_Recv(buf, numElements, MPI_INT, MPI_ANY_SOURCE, MY_MSG_TAG, MPI_COMM_WORLD);

      // can use buf here
      printf("The first element of buf is %d\n", buf[0]);
   }

   MPI_Finalize();
}
```

```
Run program using:

mpirun -np 2 ./myprogram
```

Non-blocking communication

```
int numElements = 1024;
int* buf = new int[numElements];

// ...

if (procId == 0) {
    MPI_Request req;
    MPI_Status status;

    // initiate send message to process 1
    MPI_Isend(buf, numElements, MPI_INT, 1, MY_MSG_TAG, MPI_COMM_WORLD, &req);

    // cannot modify buf here

    MPI_Wait(&req, &status);

    // send is complete, now safe to modify
    delete [] buf;
} else {
    MPI_Request req;
    MPI_Status status;

    // initiate recv message
    MPI_Irecv(buf, numElements, MPI_INT, MPI_ANY_SOURCE, MY_MSG_TAG, MPI_COMM_WORLD, &req);

    // cannot access buf yet

    MPI_Wait(&req, &status);

    // recv is complete. Now it's safe to read buf
    printf("The first element of buf is %d\n", buf[0]);
}
```

```
Run program using:

mpirun -np 2 ./myprogram
```

Non-blocking communication

```
MPI_Request req;
MPI_Status status;

// initiate recv message
MPI_Irecv(buf, numElements, MPI_INT, MPI_ANY_SOURCE, MY_MSG_TAG, MPI_COMM_WORLD, &req);

// cannot access buf yet
int flag = 0;

// loop, checking for recv completion.  If not complete, go do
// something else for awhile
while (flag == 0) {

  MPI_Test(&req, &flag, &status));

  // do other work for a bit ...
}

// now it’s safe to read buf
printf(“The first element of buf is %d\n”, buf[0]);
```

More API

- Useful higher level functionality
  - MPI_Barrier
  - MPI_BCast
  - MPI_Reduce
  - These could all be implemented manually using MPI_Send/Recv, but raising level of abstraction enables optimized implementations

- May also want to review documentation of MPI datatypes

ASSIGNMENT 2 AND EXAM 1 DISCUSSION

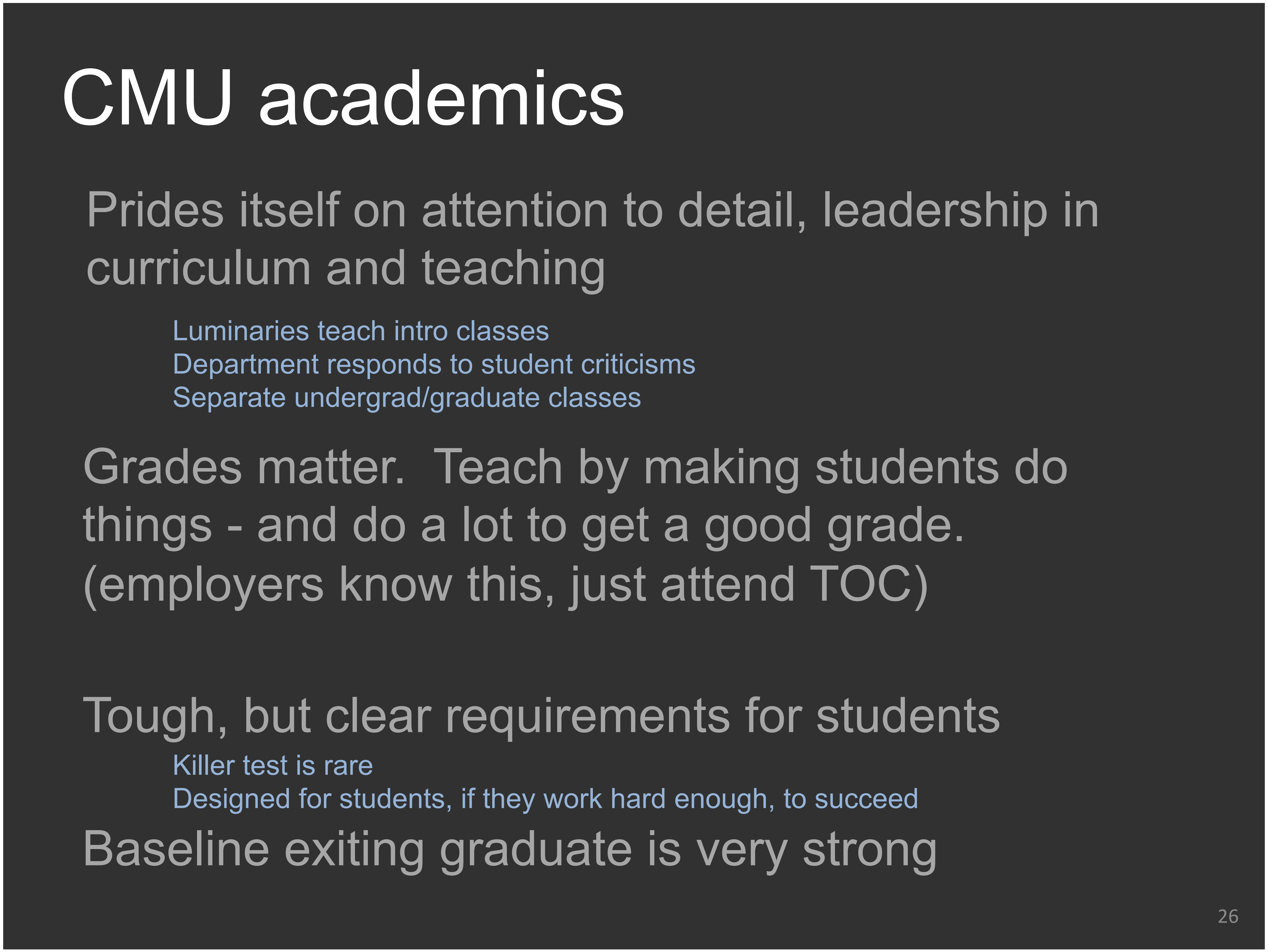
(details/solutions covered in class, not posted online)

CLASS FIRESIDE CHAT

Why do we give exams?

Why do we give projects?

What can you do to maximize your undergrad experience at CMU?



CMU academics

Prides itself on attention to detail, leadership in curriculum and teaching

- Luminaries teach intro classes
- Department responds to student criticisms
- Separate undergrad/graduate classes

Grades matter. Teach by making students do things - and do a lot to get a good grade. (employers know this, just attend TOC)

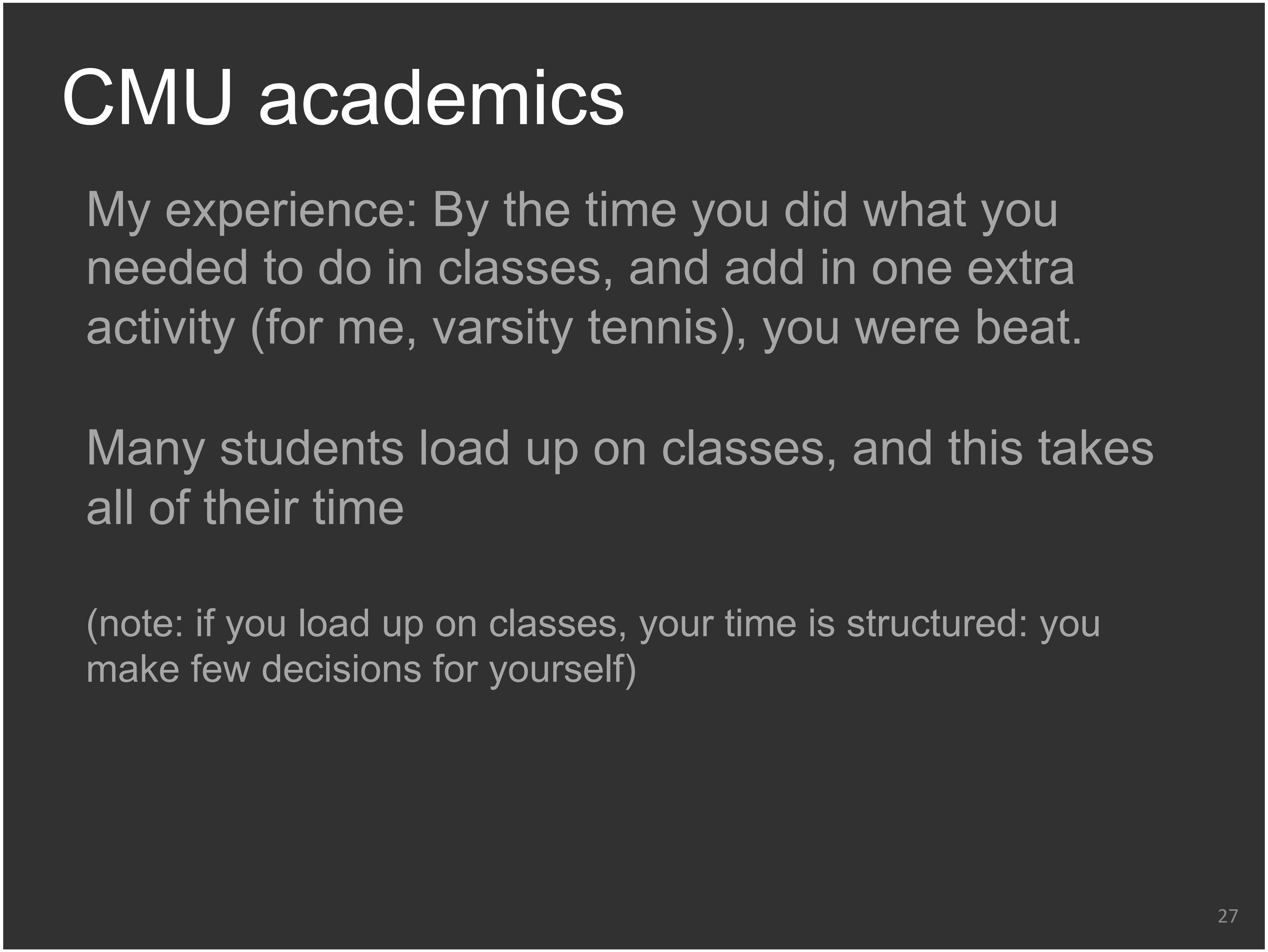
Tough, but clear requirements for students

- Killer test is rare
- Designed for students, if they work hard enough, to succeed

Baseline exiting graduate is very strong

26

GPA is meaningful at CMU



CMU academics

My experience: By the time you did what you needed to do in classes, and add in one extra activity (for me, varsity tennis), you were beat.

Many students load up on classes, and this takes all of their time

(note: if you load up on classes, your time is structured: you make few decisions for yourself)

27

Most of you are 2-14 months from graduating

- The #1 way to get into grad school is obtain a good letter of recommendation from a faculty member
  - To get the recommendation, you need to be part of a research team (or do your own research)
- What if you blew off most of this entire class, and then clearly had one of the best projects at the parallelism competition? (how would that look to a company)
  - What grade would you get?
- What if you gave modest effort in other classes, and did outstanding work on the exams, assignments, and final project in this class?
  - or conversely, gave modest effort here and went to town in another class?

HAVE A GOOD SPRING BREAK!