15-123:
Effective Programming in C and Unix

With Hunter Pitelka

Carnegie Mellon
School of Computer Science

Exam 2
and
Fun with Pointers

Recitation 9
Wednesday October 22nd, 2008

Overview

- Quickly go over the exam
- Answer questions about the exam
- Fun with Pointers

Exam2Q1:

- Come to Office hours to get these solutions!
- 7-10pm Monday-Thursday in WeH 5419D

Exam2 Q2

- Come to Office hours to get these solutions!
- 7-10pm Monday-Thursday in WeH 5419D

Exam 2, Q3

```
int readFile(char *inputFile,
             scorecard ***scorecardList,
             int *scorecardListSize);

int findhighestScorecard(scorecard **scorecardList,
                         int scorecardListSize);

int freeScorecardList(scorecard ***scorecardList,
                      int scorecardListSize);


typedef struct{
   char name[512];
   int score;
}scorecard;
```

Exam2Q3.c readFile(...);

```
int readFile(char *inputFile,
             scorecard ***scorecardList,
             int *scorecardListSize){

    FILE *fp;
    int i,number;
    char buf[512];
    scorecard *temp;

    fp = fopen(inputFile,"r");
    fscanf(fp,"%d\n",scorecardListSize);

    *scorecardList = (scorecard **) malloc(
                        (*scorecardListSize) * sizeof(scorecard *)
                    );

    for(i=0;i<*scorecardListSize;i++){
        fscanf(fp,"%s %d\n",buf,&number);
        temp = (scorecard *) malloc(sizeof(scorecard));
        strcpy(temp->name,buf);
        temp->score = number;
        *((*scorecardList)+i) = temp;
    }
    fclose(fp);
    return 0;
}
```

Exam2Q3.c findHighestScorecard(..);

```
int findhighestScorecard(scorecard **scorecardList,
                         int scorecardListSize){

    int currMax, i, currMaxIndex;
    currMax = 0;
    currMaxIndex = 0;

    for(i=0;i<scorecardListSize;i++){
        if((*(scorecardList + i))->score > currMax){
            currMax = (*(scorecardList +i))->score;
            currMaxIndex = i;
        }
    }


    return currMaxIndex;
}
```

Exam2Q3.c freeScorecardList(..);

```
int freeScorecardList(scorecard ***scorecardList,
                      int scorecardListSize){

    int i;

    for(i=0;i<scorecardListSize;i++){
        free(*(*scorecardList +i));
    }
    free(*scorecardList);
    scorecardList=NULL;

    return 0;
}
```

Exam2Q3.c main(..);

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

    scorecard ** scorecardList;
    int scorecardListSize,max;

    if(argc !=2){
        printf("Usage: %s filename\n",argv[0]);
        exit(-1);
    }

    readFile(argv[1],&scorecardList,&scorecardListSize);

    max = findhighestScorecard(scorecardList,scorecardListSize);
    printf("%s has the highest score.\n",
                    (*(scorecardList +max))->name);

    freeScorecardList(&scorecardList,scorecardListSize);

    return 0;
}
```

Fun with Pointers

Bad Memory!

```
/* return y = Ax */
int *matvec(int **A, int *x) {
   int *y = malloc(N*sizeof(int));
   int i, j;

   for (i=0; i<N; i++)
      for (j=0; j<N; j++)
         y[i] += A[i][j]*x[j];
   return y;
}
```

Dangerous Dereferences

```
int **p;

p = malloc(N*sizeof(int));

for (i=0; i<N; i++) {
   p[i] = malloc(M*sizeof(int));
}
```

Errors can occur anywhere

```
x = malloc(N*sizeof(int));
        <manipulate x>
free(x);

y = malloc(M*sizeof(int));
        <manipulate y>
free(x);
```

Lol Hunter

I made this error in recitation at the beginning of the semester.

```
int *search(int *p, int val) {

   while (*p && *p != val)
      p += sizeof(int);

   return p;
}
```

2+2=5, right?

```
int **p;

p = malloc(N*sizeof(int *));

for (i=0; i<=N; i++) {
   p[i] = malloc(M*sizeof(int));
}
```

Stack v. Heap

```
int *foo () {
   int val;

   return &val;
}
```

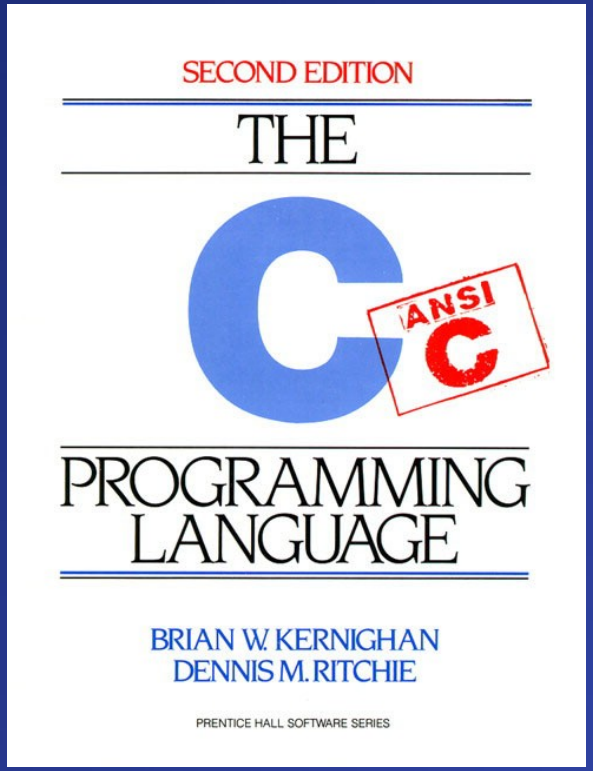
Linked List Warning

```
struct list {
    int val;
    struct list *next;
};

foo() {
    struct list *head = malloc(sizeof(struct list));
    head->val = 0;
    head->next = NULL;
    <create and manipulate the rest of the list>
    ...
    free(head);
    return;
}
```

Recitations from now on

- "How to write **effective** programs in C"
- We'll be looking at small programs that demonstrate incredibly important concepts.



Final Words

- Think about the problem before you write the solution.
- Pointers can cause you extreme pain, get used to it.

kthxbai