COS 217 Midterm Exam

Princeton University, Fall 2002

October 21, 2002

	Score	Possible
1		10
2		10
3		30
4		30
5		10
6		10
7		20
Total		120

Your name:

Unix login:

Precept number: 1 2 3

Write and sign the honor pledge:

1 English to C

Give the C declarations corresponding to the English type descriptions. For example, "f is a 3-element array of integers" translates to `int f[3];`.

1. va is a 4-element array of pointers to int.

2. vb is a 3-element array of pointers to 4-element arrays of int.

3. vc is pointer to function taking void and returning void

4. vd is 2-element array of pointers to functions taking an int and returning a char.

5. ve is a pointer to a function taking a pointer to a char and returning a pointer to a function taking an int array and returning a pointer to char.

2 C to English

Translate the following C declarations into English.

1. `float * a[5];`

2. `char (*b)[2][3];`

3. `int * (*f)(char x, int);`

4. `char * f(int * x, float (*y)[]);`

5. `char (*g(int x))(int y);`

3 Fix the bugs

The following program is meant to find anagrams. For each word (such as "beaters") it sorts the letters into alphabetical order (such as "abeerst") and enters the pair in a table. Since anagrams (such as "beaters" and "berates") have the same key ("abeerst"), one can find anagrams by looking up in the table. The program is supposed to produce correct output as long as the words aren't too long and there aren't too many words, and it's not supposed to crash on any input.

For an input file such as,

```
abacus
abbess
accent
alert
alexia
alter
beaters
beauty
berates
```

the output of the program is supposed to be,

```
alter is an anagram of alert
berates is an anagram of beaters
```

However, there are 7 bugs in the program. Correct the bugs.

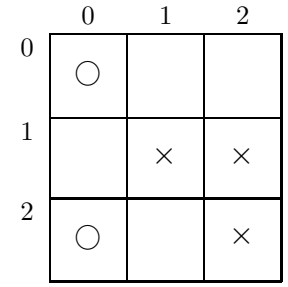
(Note: gcc doesn't notice any problem with this program; it's syntactically correct.)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TABLESIZE 10000

void swap(char a, char b) {
  char t = a;
  a = b; b = t;
}

char *sort(char *s) {
  char *p; int i,j;
  int n = strlen(s);
  for (i=0; i<n; i++)
    p[i]=s[i];
  for (i=0; i<n; i++)       /* no bugs in this line */
    for (j=0; j<i; j++)     /* no bugs in this line */
      if (p[j]>p[j+1])      /* no bugs in this line */
       swap(p[j], p[j+1]);
  return p;
}

struct entry {char *key, *value;};
struct entry table[TABLESIZE];
static int numentries=0;

void addToTable(char *key, char *value) {
  table[numentries].key=key;
  table[numentries].value=value;
  numentries++;
}
```

```
/* Find all pairs of entries in the table that sort to the same thing. */
void printAnagrams() {
  int i,j;
  for(i=0; i<numentries; i++)
    for(j=0; j<i; j++)
      if (table[i].key == table[j].key)
        printf("%s is an anagram of %s\n",
               table[i].value, table[j].value);
}

int main (int argc, char **argv) {
  char buf[100];
  while (gets(buf)) {
    addToTable(sort(buf), buf);
  }
  printAnagrams();
  /* don't worry about calls to free() for this exam problem */
  return 0;
}
```

4 Implement an interface

Consider an ADT for a Tic-Tac-Toe board:



The interface for this ADT is as follows:

```
/* tictac.h */

typedef int player;  /* values are None, X, O */
#define None 0
#define X 1
#define O 2

typedef struct board *Board_T;

Board_T  Board_new(void);

player  Board_whoseTurnIsIt(Board_T b);
/* return 'None' if the game is over. */

void Board_makeMove(Board_T b, player p, int row, int col);

void Board_unMakeMove(Board_T b, player p, int row, int col);

player Board_whatIsInCell(Board_T b, int row, int col);

double Board_eval(Board_T b);

void Board_foreach(Board_T b,
                   void (*f)(int row, int col, player p, void *extra),
                   void *extra);
```

Your task is to implement

- The data structure for the representation of a `Board_T`.
- The `Board_whoseTurnIsIt` function.
- The `Board_foreach` function, which is supposed to call `f` on every square, passing it the location and contents of the square.
- A client function `countTheXes` which takes a board and uses `Board_foreach` to count how many squares are occupied by an X.

For this exam, don't worry about writing assertions.

```
/* tictac.c */
#include <stdio.h>
#include <stdlib.h>
#include "tictac.h"

/* data structure for representation of a Board_T */









/* Function to tell whose turn it is. */
player  Board_whoseTurnIsIt(Board_T b) {









}


/* Function to apply a function to the contents of every square */
void Board_foreach(Board_T b,
                   void (*f)(int row, int col, player p, void *extra),
                   void *extra) {









}
```

```c
/* client.c */
#include <stdio.h>
#include <stdlib.h>
#include "tictac.h"









int countTheXes(Board_T b) {









}
```

5 Scope, linkage, and duration

Consider the following program. Note that it is implemented in two files. What will the computer print when it executes the program? Draw a box around your answer.

File one.c:

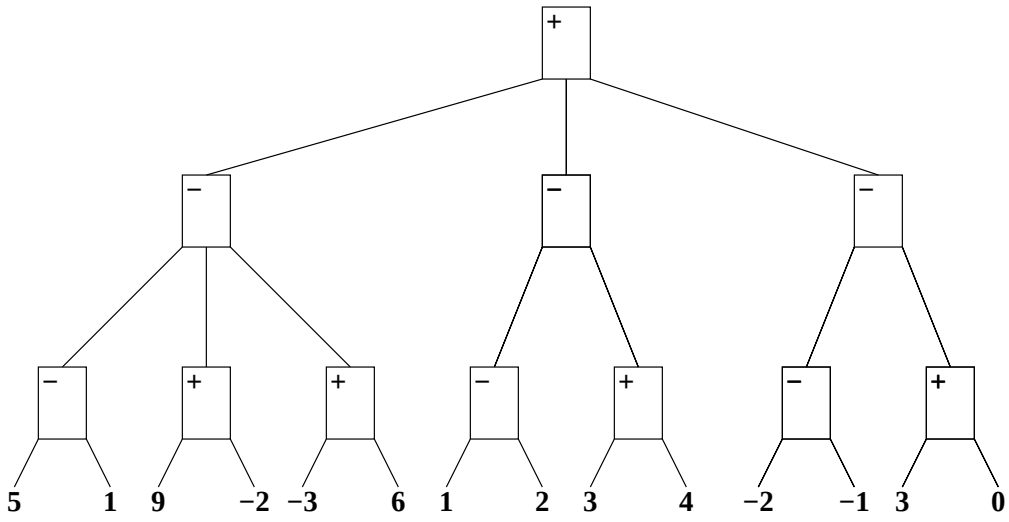
```
#include <stdio.h>

int a;
int b;
int c;

void f(int);

int main(void)
{
   a = 10;
   b = 100;
   c = 1000;

   printf("%d %d %d\n", a, b, c);
   f(c);
   printf("%d %d %d\n", a, b, c);
   a++; b++; c++;
   f(c);
   printf("%d %d %d\n", a, b, c);

   return 0;
}
```

File two.c:

```
#include <stdio.h>

extern int a;
static int b;

void f(int c)
{
   int d = 10000;
   static int e = 100000;
   printf("%d %d %d %d %d\n",
            a, b, c, d, e);
   a++; b++; c++; d++; e++;
}
```

6 Alpha-beta search

Below is a game tree, with the heuristic values shown at depth 4. Perform alpha-beta search on this tree. In each box *that is traversed by the search*, show the value computed for that node. Indicate alpha-beta cutoffs by putting an X over any edge leading to a subtree not searched.



7 Code re-use

Suppose you and your team want to modify your Kalah player to play some other game, such as Chess or Checkers. What modifications are necessary?

Write your team color here:

A. Find (attached) the header files for your team. Circle each line of code in any of the header files that would have to be different for playing another game. Also circle any portion of any comment that would have to be different.

B. Find (attached) the list of .c files for your team. Circle the name of each .c file that would have to be modified if you played a different game. Next to the file name, write a *brief* explanation (10 words or less) why it has to be modified, or why not.

Hint: in just 10 words, you don't always need complete sentences.