ECE 264-2 Final Exam

3:20-5:20PM, May 8, 2010

I certify that I will not receive nor provide aid to any other student for this exam.

Signature:

You must sign here. Otherwise, you will receive **1-point** *penalty.*

Did you fill the on-line course evaluation before May 2?

Yes No

This exam is printed **double sides**. Please read the questions carefully. Two common mistakes are answering wrong questions and failing to answer all questions.

This is an *open-book, open-note* exam. You can use any book or note or program printouts.

Please turn off your cellular phone and iPod. No electronic device is allowed.

Contents

Total Score: out of 21.

This page is blank. You can write answers here.

1 Recursion (4 points)

Rewrite the following code to a **non**-recursive function.

```
#include <stdio.h>
#include <stdlib.h>
int rec1(int n, int k)
{
  if (n < k)  { return 0; }
  if (k == 0) { return 1; }
  return 3 * rec1(n - 1, k - 1) + 2 * rec1(n - 1, k);
}

int rec2(int n, int k)
{
  /* do not use recursion */
  int * * val; /* array of dimension (n + 1) x (k + 1) */
  int i, j; /* counters upto n and k */
  int result;
  /* declare additional local variables, if necessary */


  /* check termination conditions (1 point) */


  /* allocate memory */
  val = malloc((n + 1) * sizeof (int));
  for (i = 0; i <= n; i ++)
    { val[i] = malloc((k + 1) * sizeof(int)); }
  /* initialize val[i][j] (1 point) */
```

```
  /* compute the answer without recursion (2 point) */
```

```
  result = val[n][k];
  /* release memory */
  for (i = 0; i <= k; i ++)
    { free(val[i]); }
  free (val);
  return result;
}
```

2 Binary Search Tree (2 points)

Draw the binary tree as numbers are inserted.

```
insert 9
insert 3
insert 11
insert 2
insert 1
insert 22
==> draw tree
```

```
continue to add more nodes
insert 19
insert 6
insert 17
insert 34
insert 7
insert 5
==> draw tree
```

3 Arithmetic Evaluation (3 point)

Draw the evaluation tree for the following arithmetic expression

```
(1 + 2) * (3 * (4 + 5)) - (6 + 7)
```

4 Complexity (4 points)

```
#include <stdio.h>
void f(int x, int * c)
{
  int i;
  (*c) ++;
  if (x == 0)
    { return; }
  for (i = 0; i < x; i ++)
    {
      f(i, c);
    }
}
int main(int argc, char * argv[])
{
  int c = 0;
  f(1, & c);
  printf("L1: c = %d\n", c);

  c = 0;
  f(2, & c);
  printf("L2: c = %d\n", c);

  c = 0;
  f(3, & c);
  printf("L3: c = %d\n", c);

  /* suppose n is a positive integer */
  c = 0;
  f(n, & c);
  printf("L4: c = %d\n", c);

  return 0;
}
```

What is the value of `c` ?

- at `L1`? (1 point).
- at `L2`? (1 point).
- at `L3`? (1 point).
- at `L4` for a positive integer `n`? (1 point).

5 Binary Search (4 points)

Write a recursive function for binary search.

```
int search(int * a, int n, int v, int h, int t)
/* a: an arry to be searched, sorted in ascending order
   n: number of elements in the array
   v: value to search
   [h, t]: range of index to search

   return 0 if v is not an element of a
   return 1 if v is an element of a
   3 points (Caution: one point in the main function)
*/
{
}
int main(int argc, char * argv[])
{
  int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14};
  int n = sizeof(a) / sizeof(int);
  /* how to call search to find whether 4 is an element of a? */
  /*
    <=== 1 point ===>
  */
  printf("search 4 = %d\n", search(                            ));
  return 0;
}
```

6 Integer Partition (4 points)

Write a function to compute the number of ways to partition a positive integer *n* so that the numbers used in each partition are equal or increasing. For example, if *n* is 4, the following are counted

```
1 + 1 + 1 + 1
1 + 1 + 2
1 + 3
2 + 2
4
```

but the following are **not** counted

```
1 + 2 + 1
2 + 1 + 1
3 + 1
```

```
type? f(int n, int * c, additional arguments?)
{
  /* *c stores the value of the number of ways to partition n */
















}
int main(int argc, char * argv[])
{
  int c = 0;
  int n = 20;
  f(n, & c, ...);
  printf("There are %d ways to partition %d\n", c, n);
  return 0;
}
```