ECE 264-1 Exam 1

01:30-2:20PM, February 24, 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.*

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.

To pass outcome 1, you need to obtain 50% or more points in the question.

Contents

Total Score: out of 20.

Passed Outcome 1: Yes No

This page is blank. You can write answers here.

1 File (8 points, outcome 1)

Write a program that reads one file (input), replaces upper-case letters (A-Z) by lower-case letters (a-z), and writes to another file (output). If the input file contains a character that is not an upper-case letter copy the character to the output file without changing it.

Reference:

```
int fputc ( int character, FILE * stream );
```

Write character to stream

Writes a character to the stream and advances the position indicator. The character is written at the current position of the stream as indicated by the internal position indicator, which is then advanced one character.

Parameters

character

Character to be written. The character is passed as its int promotion.

stream

Pointer to a FILE object that identifies the stream where the character is to be written.

Return Value

If there are no errors, the same character that has been written is returned. If an error occurs, EOF is returned and the error indicator is set (see ferror).

```
int fgetc ( FILE * stream );
```

Get character from stream

Returns the character currently pointed by the internal file position indicator of the specified stream. The internal file position indicator is then advanced by one character to point to the next character. fgetc and getc are equivalent, except that the latter one may be implemented as a macro.

Parameters

stream

Pointer to a FILE object that identifies the stream on which the operation is to be performed.

Return Value

The character read is returned as an int value. If the End-of-File is reached or a reading error happens, the function returns EOF and the corresponding error or eof indicator is set. You can use either ferror or feof to determine whether an error happened or the End-Of-File was reached.

```
#include <stdio.h>
int main(int argc, char * argv[])
/*
  The program needs two additional arguments:
  input (first) and output (second) file names
 */
{
  /* Check whether two file names are given.
     If not, print an error message and terminate. 1 point */




  /* open the input and output files. 1 point */




  /* check whether the openings succeed.
     If not, print an error message and terminate. 1 point */




  /* read the input file. 1 point

     check whether a character is between A and Z. 1 point

     if it is, replace it by the lower case letter
     if it is not, do not change it. 1 points

     write to the ouput file. 1 point  */







  /* close the files. 1 point */



  return 0;
}
```

2 Sorting (4 points)

Write down the values of

- `cnt1` (1 point)
- `cnt2` (1 point)
- `cnt3` (1 point)

after running the code.

Is the sorted array in the ascending or the descending order? (1 point)

```
#include <stdio.h>
void swap(int * a, int * b)
{
  int temp = *a;
  (*a) = (*b);
  (*b) = temp;
}

int main(int argc, char * argv[])
{
  int x[] = {6, 7, 8, -11, 5, 20, 2, 30};
  int n = sizeof(x) / sizeof(int);
  int i1, i2, mInd;
  int cnt1 = 0;
  int cnt2 = 0;
  int cnt3 = 0;
  for (i1 = 0; i1 < n - 1; i1 ++)
    {
      mInd = i1;
      for (i2 = i1 + 1; i2 < n; i2 ++)
        {
          cnt1 ++;            /* change cnt1 */
          if (x[mInd] > x[i2])
            {
              cnt2 ++;        /* change cnt2 */
              mInd = i2;
            }
        }
      if (mInd != i1)
        {
          cnt3 ++;            /* change cnt3 */
          swap(&x[i1], &x[mInd]);
        }
    }
  printf("n = %d, counters = %d %d %d\n", n, cnt1, cnt2, cnt3);
  return 0;
}
```

3 Control (4 points)

3.1 `if` (2 points)

Consider the following code

```
int x = 3;
int y = 5;
...
/* code that may modify x and y */
...
/* L1 */
if (x == 0)
{
   y = 0;
}
/* L2 */
```

Suppose there is no statement between `L1` and the following `if`; neither is any statement between `L2` and the }. Which statement (or statements) must be true? You may choose multiple answers.

A. If `y` is 0 at `L2`, `x` must be 0 at `L1`.

B. If `x` is 1 at `L1`, it is impossible that `y` is 0 at `L2`.

C. If `y` is 0 at `L1`, `y` must be zero at `L2`.

D. If both `x` and `y` are 1 at `L1`, `y` must be zero at `L2`.

E. If `y` is 1 at `L2`, `x` must **not** be 0 at `L1`.

3.2 `switch-case` (2 points)

What are the values of `x` and `y` after executing the following code?

```
#include <stdio.h>
int main(int argc, char * argv[])
{
  int w = 0;
  int x = 0;
  int y = 0;
  int z = 0;
  switch (w)
    {
    case 0:
      if (x == 0)
        { y = 3; }
      else
        { z = 4; }
      /* no break */
    case 1:
      if (y > 0)
        {
          if (z != 0)
            { x = 2; }
          else
            { x = 1; }
        }
      else
        { w = -5; }
      w = -6;
      break;
    case 2:
      w = 9;
      x = 8;
      break;
    default:
      y = 7;
      z = 10;
    }
  printf("%d %d %d %d\n", w, x, y, z);
  return 0;
}
```

4 Pointer (4 points)

Write a function that can swap two strings and call the function. The `swap` function should **not** assume that the two input strings have the same length. If your program allocates memory, make sure your program does not leak memory.

```
/*
 * fill the parts marked by =====
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void swap(/*==== arguments? 1 point ====*/ )
{

  /*==== how to swap? 2 points ====*/

}

int main(int argc, char * argv[])
{
  char * s1;
  char * s2;
  s1 = malloc(80 * sizeof(char));
  s2 = malloc(80 * sizeof(char));
  strcpy(s1, "purdue");
  strcpy(s2, "ece");
  printf("%s\n", s1);           /* purdue */
  printf("%s\n", s2);           /* ece */

  swap(/*==== how to call? 1 point ====*/);

  printf("%s\n", s1);           /* ece */
  printf("%s\n", s2);           /* purdue */
  free (s1);
  free (s2);
  return 0;
}
```