Programming Languages (CS302 2007S)

Homework 2: Permutations

Assigned: Friday, January 26, 2007
Due: Friday, February 2, 2007
No extensions!

Summary: In this assignment, you will reimplement a classic (and goto-laden) algorithm for computing permutations.

Purposes: To give you more experience understanding gotos. To help you think about the relationships between unstructured and structured code.

Expected Time: Two to three hours.

Collaboration: You may work alone or you may work in a group of size two or three. I encourage you to work in groups on this assignment. You may discuss the assignment with anyone you'd like, provided you cite such discussions in your assignment.

Submitting: Email me your answer. More details below.

Warning: So that this exercise is a learning assignment for everyone, I may spend class time publicly critiquing your work.

Assignment

Read (or reread) Bratley's Permutations with Repetitions and then

1. Reimplement the algorithm in C, staying as close to the original structure as is possible. Call the procedure `perm` and store it in the file `perm.c`.

2. Test your implementation. Here's a program that might help you get started in testing. I'd recommend saving it as `testperm.c`

```
#include <stdio.h>
#include <stdlib.h>

extern void perm(int[], int, int *);

void printa(int a[], int n)
{
  int i;
  printf("a = [%d", a[0]);
  for (i = 1; i < n; i++)
    printf(", %d", a[i]);
  printf("]\n");
} /* printa(int[], int) */

int main(int argc, char *argv[])
```

```
{
  int last = 1;
  int i;

  /* Build the array from the command line. */
  int n = argc-1;
  int *a = malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
    a[i] = atoi(argv[i+1]);

  /* Repeatedly permute and print. */
  perm(a, n, &last);
  while (!last) {
    printa(a, n);
    perm(a, n, &last);
  } /* while (!last); */
} // main
```

You can then create a tester with

```
cc -c testperm.c -o testperm.o
cc -c perm.c -o perm.o
cc testperm.o perm.o -o testperm
```

3. Rewrite `perm` so that it no longer has gotos. You'll need loops, conditionals, and the ilk. (Your code may also be longer.) You might store your rewritten code in the file `newperm.c`.

4. Explain the steps involved in the rewrite.

Important Evaluation Criteria

Students who correctly translate the algorithm into C (parts 1 and 2) will earn a check. Students who complete parts 3 and 4 will earn a plus. Students who cannot correctly translate the algorithm will earn a lower score.

Submitting Your Homework

Please submit this via email, using a title of CSC302 HW2.

Attach the files that you've created to that email message.