CS 1723, Data Structures
Fall Semester, 1998
Second Examination

1. (a) Consider the following arithmetic expression, similar to ones we used in a programming assignment (the # just terminates the expression):

   `2+3*4#`

   Give the RPN form of this expression. What rule in C tells us that the `*` is performed before the `+`?

   (b) There is a significant difference between the expressions

   `4*3*2#` and `4^3^2`

   Describe the difference and give the corresponding RPN.

2. Consider the following C code, which reads "words" from `stdin` and inserts them onto a linked list. Supply the code for the missing functions `printlist`, `printreverse` and `insert`. `printlist` should just chase down the list, printing words as it encounters them. `printreverse` should use *recursion* to print the list in reverse order (the order in which they were read). The `insert` function should insert successive words into the list pointed to by `p`. (I assume that `insert` puts each new node at the head of the list. Hints about `insert` are included below.)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct lnode {
   char *word;
   struct lnode *next;
};
struct lnode *insert(char *w, struct lnode *p);
void printlist(struct lnode *p);
void printreverse(struct lnode *p);

void main(void)
{
   struct lnode *p = NULL;
   char w[100];
   while (scanf("%s", w) != EOF)
      p = insert(w, p);
   printlist(p);
   printreverse(p);
   printf("\n");
}

void printlist(struct lnode *p)
{
   /* code for printlist here */
   printf("\n");
}
```

```
void printreverse (struct lnode *p)
{
   /* code for printreverse here (no loops) */
}
struct lnode *insert(char *w, struct lnode *p)
{
   struct lnode *r;
   /* Set r = malloced storage for an lnode.                 */
   /* Set r -> word = malloced storage for the input string. */
   /* Copy the input string into the new storage.            */
   /* Give r -> next the proper value.                       */
   /* Return the proper pointer.                             */
}
runner% cc -o exam2_2 exam2_2.c
runner% exam2_3
Now is the time for   (ctrl-d entered to simulate EOF)
for time the is Now   (printed by printlist)
Now is the time for   (printed by printrecurse)
```

3. Consider the following C program, with parts omitted:

```
/* Weeks program */
#include <stdio.h>

void main(void)
{
   int i;
   char *weeks[] = {"none", "Mon", "Tue",
      "Wed", "Thu", "Fri", "Sat", "Sun", NULL};
   char **p;
   for(i = 1; i < 8; i++)
      printf("%s  ", /* PART (a) */);
   printf("\n");
   for(i = 1; i < 8; i++)
      printf("%s  ", /* PART (b) */);
   printf("\n");
   p =  /* PART (c) */;
   while ( /* PART (d) */)
      printf("%s  ",  /* PART (e) */);
   printf("\n");
}
runner% exam2_3
Mon  Tue  Wed  Thu  Fri  Sat  Sun    (repeat line twice more)
```

(a) Supply an expression involving weeks and `[]` that will print correctly.

(b) Supply an expression involving weeks *without* `[]` that will print correctly.

(c) Supply an expression involving `p` that will position the pointer to skip the first entry.

(d) Supply an expression involving `p` that will terminate with the `NULL` at the array's end.

(e) Supply an expression involving `p` that will print correctly. (Notice that you must change `p` here; the variable `i` is not part of this loop.)