

PROBLEM 1.

```
runner% cat final1.c
#include <stdio.h>
```

```
void main(void)
{
    int i = 0, n;
    scanf("%i", &n);
    for (i = n-1; i >= 0; i--)
        printf("%i ", i);
    printf("\n");
}
coyotel9% cc -o final1 final1.c
coyotel9% final1
5
4 3 2 1 0
runner%
```

```
/* another solution */
#include <stdio.h>
```

```
void main(void)
{
    int n;
    scanf("%i", &n);
    while (n > 0)
        printf("%i ", i);
    printf("\n");
}
```

PROBLEM 2.

```
#include <stdio.h>
void main(void)
{
    int i;
    double arr[200];
    int zero_found;
    for (i = 0; i < 100; i++)
        arr[i] = (double)(i+1);
    /* search for zero */
    zero_found = 0;
    for (i = 0; i < 100; i++) {
        if (arr[i] == 0.0) zero_found = 1;
        break; /* works without the break */
    }
    if (zero_found) printf("Zero found\n");
    else printf("No zeros\n");
}
```

PROBLEM 3.

```
#include <stdio.h>
#include <ctype.h>
void main(void)
{
    char ch;
    int digits = 0;
    while( (ch = getchar()) != EOF)
        if (isdigit(ch)) digits++;
    printf("Number of digits: %i\n", digits);
}
runner% cc -o final_1 final_1.c
runner% final_1 < final_1.c
Number of digits: 1
```

PROBLEM 4.

```
runner% cat final3.c
#include <stdio.h>
#define MAXFIB 40

void main(void)
{
    int f[MAXFIB];
    int i;
    f[0] = 0; f[1] = 1;
    for (i = 2; i < MAXFIB; i++)
        f[i] = f[i-1] + f[i-2];
    for (i = 0; i < MAXFIB; i++)
        printf("Fibonacci number %2i = %8i\n",
            i, f[i]);
}

runner% cc -o final3 final3.c
```

```
runner% final3
Fibonacci number 0 =      0
Fibonacci number 1 =      1
Fibonacci number 2 =      1
Fibonacci number 3 =      2
Fibonacci number 4 =      3
Fibonacci number 5 =      5
... (lines missing)
Fibonacci number 38 = 39088169
Fibonacci number 39 = 63245986
```

PROBLEM 5. Carnivals love this game because they make an average of 7.87 cents for each dollar that players bet. The players ("suckers") sometimes reason as follows: there are 3 chances out of 6 that my number will come up, so I should come out even that way, and the extra paid off for duplicates is just gravy.

```
/* Simulate game of Chuck-O-Luck
 * Each time, read in a number to bet on: bet_on, 1 <= bet_on <= 6
 * Simulate rolling three dice, results in: d[0], d[1], d[2]
 * Count number of dice matching your number: num_dice
 * If num_dice > 0, you win num_dice dollars, but if
 *   num_dice == 0, you lose a dollar.
 * Program written by N. Wagner, April 29, 1998
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_GAMES 100000 /* total number of games to play */
int roll(void); /* single roll of a die */
void main(void)
{
    int bet_on = 4; /* the number you are betting on */
    int winnings = 0; /* total winnings for all games */
    int win[4] = {0, 0, 0, 0}; /* keep track of the number of times the
    /* dice rolled matched your number */
    int n; /* count the games */
    int i; /* counter used for dice */
    int num_dice; /* number of dice matching your number */
    int d[3]; /* results of three rolls */
    srand48((long)time(NULL)); /* initialize random number generator */
    for (n = 0; n < MAX_GAMES; n++) { /* play MAX_GAMES number of games */
        d[0] = roll(); d[1] = roll(); d[2] = roll(); /* three rolls */
        /* count number of rolls matching your number */
        num_dice = 0;
        for (i = 0; i < 3; i++)
            if(d[i] == bet_on) num_dice++;
        win[num_dice]++; /* keep track of each value of num_dice */
        if (num_dice == 3) winnings += 3; /* add to winnings */
        else if (num_dice == 2) winnings += 2; /* add to winnings */
        else if (num_dice == 1) winnings += 1; /* add to winnings */
        else if (num_dice == 0) winnings += -1; /* take from winnings */
    }
    printf("Total winnings:%i dollars, or %.3f%%", winnings,
        100.0*winnings/MAX_GAMES);
    printf(", out of %i bet total\n", MAX_GAMES);
    printf("Percentages: 3: %.5f%%, 2: %.5f%%, 1: %.5f%%, 0: %.5f%%\n",
        win[3]*100.0/MAX_GAMES, win[2]*100.0/MAX_GAMES,
        win[1]*100.0/MAX_GAMES, win[0]*100.0/MAX_GAMES);
    printf("Exact values: 3: %.5f%%, 2: %.5f%%, 1: %.5f%%, 0: %.5f%%\n",
        1.0*100.0/216.0, 15.0*100.0/216.0,
        75.0*100.0/216.0, 125.0*100.0/216.0);
}
int roll(void)
{
    return (int)(6.0*drand48() + 1.0);
}
runner% final (with MAX_GAMES equal to 100000000)
Total winnings:-7855374 dollars, or -7.855%, out of 100000000 bet total
Percentages: 3: 0.46376%, 2: 6.94421%, 1: 34.72848%, 0: 57.86355%
Exact values: 3: 0.46296%, 2: 6.94444%, 1: 34.72222%, 0: 57.87037%
Notice that you are losing nearly 8% of the money you bet. (7.87% on the average.)
If we make the one change that the game pays $20 in case of all three dice are equal
to your number, then the game is exactly fair.
runner% final (with MAX_GAMES equal to 100000000 and one line of
code changed: if (num_dice == 3) winnings += 3;
changed to: if (num_dice == 3) winnings += 20;)
Total winnings:3990 dollars, or 0.004%, out of 100000000 bet total
Percentages: 3: 0.46306%, 2: 6.94374%, 1: 34.72425%, 0: 57.86894%
Exact values: 3: 0.46296%, 2: 6.94444%, 1: 34.72222%, 0: 57.87037%
```