

Solutions to 1997-98 Midterm

Problem 1: Karel the Robot (10 points)

There are several different strategies for solving this problem. The following solution is one possibility:

```
/*
 * File: trick-or-treat.c
 * -----
 * This program solves the Trick-or-Treat problem from the midterm
 * exam.
 */

#include "karel.h"
#include "turns.h"

/* Procedure prototypes */

procedure TrickOrTreat();
procedure CollectOneSideOfStreet();
procedure CollectOneHouse();
procedure CollectTreat();

/* Main program */

main()
{
    CollectOneSideOfStreet();
    TurnAround();
    CollectOneSideOfStreet();
}

/*
 * Procedure: CollectOneSideOfStreet
 * -----
 * This procedure walks down the street looking at the houses
 * on Karel's right.
 */

procedure CollectOneSideOfStreet()
{
    while (FrontIsClear()) {
        CollectOneHouse();
        Move();
    }
    CollectOneHouse();
}
```

```
/*
 * Procedure: CollectOneHouse
 * -----
 * This procedure first checks to see if Karel has a doorway
 * to its right.  If so, Karel enters the porch area, picks
 * up any treat that is there, and comes back out of the doorway
 * to end up facing in the original direction.
 */

procedure CollectOneHouse()
{
    if (RightIsClear()) {
        TurnRight();
        Move();
        CollectTreat();
        TurnAround();
        Move();
        TurnRight();
    }
}

/*
 * Procedure: CollectTreat
 * -----
 * This procedure collects a beeper if any exist on the current
 * corner.
 */

procedure CollectTreat()
{
    if (BeeperPresent()) {
        PickBeeper();
    }
}
```

Problem 2: Simple C expressions, statements, and functions (10 points)

(2a) Type: `double`
Value: 10.31

The answer, for those who didn't notice, was the date of the exam (October 31).

(2b) Value: 7

On successive cycles of the loop, the variables `k` and `n` take on the following values:

```
k = 0  n = 3
k = 1  n = 10
k = 2  n = 5
k = 3  n = 16
k = 4  n = 8
k = 5  n = 4
k = 6  n = 2
k = 7  n = 1
```

The question of whether this function halts for all positive integers is an unsolved question in mathematics. The list of values generated by this function for a given input is variously called the Ulam sequence or the hailstone sequence.

(2c) Output:

```
s1: Halloween
s2: HALLOWEEN-31
```

Problem 3: Simple C programs (15 points)

```
/*
 * File: toptwo.c
 * -----
 * This program reads in a list of nonnegative integers
 * and displays the two largest values.  The end of the
 * list is indicated by a sentinel, which is defined in
 * the "Constants" section.
 */

#include <stdio.h>
#include "genlib.h"
#include "simpio.h"

/*
 * Constants
 * -----
 * Sentinel -- Value that terminates the input list
 */

#define Sentinel 0

/* Main program */

main()
{
    int largest, secondLargest, current;

    printf("This program finds the two largest integers in\n");
    printf("a list.  Use %d to end the input.\n", Sentinel);
    largest = secondLargest = -1;
    while (TRUE) {
        printf(" ? ");
        current = GetInteger();
        if (current == Sentinel) break;
        if (current > largest) {
            secondLargest = largest;
            largest = current;
        } else if (current > secondLargest) {
            secondLargest = current;
        }
    }
    printf("Largest value:  %d\n", largest);
    printf("Second largest: %d\n", secondLargest);
}
}
```

Problem 4: Using the graphics library (15 points)

```

/*
 * File: checkers.c
 * -----
 * This program draws an 8x8 checkerboard initialized to the
 * standard starting configuration. The lower left corner of
 * the board appears at the origin of the graphics window.
 */

#include <stdio.h>
#include "genlib.h"
#include "graphics.h"

/*
 * Constants
 * -----
 * SquaresOnSide -- Squares along each side of the board
 * SquareSize    -- Size of each square in inches
 * CheckerRadius -- Radius of circular checker
 */

#define SquaresOnSide 8
#define SquareSize    0.35
#define CheckerRadius 0.10

/* Function prototypes */

void DrawCheckerboard(double x, double y);
void DrawSquare(double x, double y, bool isOccupied);
void DrawCheckerInSquare(double x, double y);
bool IsInitiallyOccupied(int col, int row);
void DrawBox(double x, double y, double width, double height);
void DrawCenteredCircle(double x, double y, double r);

/* Implementation */

main()
{
    InitGraphics();
    DrawCheckerboard(0, 0);
}

/*
 * Function: DrawCheckerboard
 * Usage: DrawCheckerboard(x, y);
 * -----
 * This function draws an initialized checkerboard at (x, y).
 */

void DrawCheckerboard(double x, double y)
{
    int row, col;

    for (row = 0; row < SquaresOnSide; row++) {
        for (col = 0; col < SquaresOnSide; col++) {
            DrawSquare(x + col * SquareSize,
                       y + row * SquareSize,
                       IsInitiallyOccupied(col, row));
        }
    }
}

```

```

/*
 * Function: DrawSquare
 * Usage: DrawSquare(x, y, pieceFlag);
 * -----
 * This function draws a single square of the checkerboard at
 * position (x, y). The pieceFlag flag indicates whether a
 * piece should be drawn in the square.
 */

void DrawSquare(double x, double y, bool pieceFlag)
{
    DrawBox(x, y, SquareSize, SquareSize);
    if (pieceFlag) DrawCheckerInSquare(x, y);
}

/*
 * Function: DrawCheckerInSquare
 * Usage: DrawCheckerInSquare(x, y);
 * -----
 * This function draws a circular checker in the square whose
 * corner at position (x, y).
 */

void DrawCheckerInSquare(double x, double y)
{
    DrawCenteredCircle(x + SquareSize / 2, y + SquareSize / 2,
                      CheckerRadius);
}

/*
 * Function: IsInitiallyOccupied
 * Usage: if (IsInitiallyOccupied(col, row)) . . .
 * -----
 * This function returns TRUE if the square would be occupied in
 * the initial configuration of a checkerboard. The problem on
 * the exam allows you to assume that the board size is 8, which
 * simplifies the coding. Here, the central rows are taken to be
 * SquaresOnSide / 2 and (SquaresOnSide - 1) / 2, so that it
 * works a board of any size.
 */

bool IsInitiallyOccupied(int col, int row)
{
    bool isCentralRow, isEvenSquare;

    isCentralRow = (row == SquaresOnSide / 2)
                  || (row == (SquaresOnSide - 1) / 2);
    isEvenSquare = ((row + col) % 2 == 0);
    return (isEvenSquare && !isCentralRow);
}

/* The remaining functions are given in the text */

```

Problem 5: Using the string, random, and scanner libraries (10 points)

```
string OrdinalForm(int n)
{
    string suffix;

    switch (n % 100) {
        case 11: case 12: case 13:
            suffix = "th";
            break;
        default:
            switch (n % 10) {
                case 1: suffix = "st"; break;
                case 2: suffix = "nd"; break;
                case 3: suffix = "rd"; break;
                default: suffix = "th"; break;
            }
    }
    return (Concat(IntegerToString(n), suffix));
}
```