

Spring 2000 CSE 142 Final Version A

Answer Key

1. D
2. A
3. D
4. C
5. C
6. D
7. E
8. A
9. D
10. D
11. C
12. D
13. D
14. B
15. B

16. See the following program:

```
/* college years */
#define FRESHMAN 0
#define SOPHOMORE 1
#define JUNIOR 2
#define SENIOR 3

typedef struct
{
    int    sid;        /* student identification number */
    char   name[50];   /* name of student */
    int    year;       /* college year of student */
    double gpa;        /* grade point average of student */
} Student;

/* Given an array student with nstudents records, print the ID */
/* number and name of each student whose gpa is greater than or */
/* equal to gpa. The names and ID numbers should be sorted by */
/* year, with seniors being printed first. */
void printHonorRoll(Student student[], int nstudents, double gpa)
{
    int year, i;

    for (year = SENIOR; year >= FRESHMAN; year--)
        for (i = 0; i < nstudents; i++)
            if (student[i].year == year && student[i].gpa >= gpa)
                printf("%10i %s\n", student[i].sid, student[i].name);
}
```

17a/b. See the following program:

```
int appearsIn(char ch, char str[])
{
    int k;
    for (k=0; str[k]!='\0'; k++)
        if (ch == str[k])
            return (1);
}
```

```

    return (0);
}

int sameLetters(char str1[], char str2[])
{
    int j, found;
    /* See if all characters of str1 are present in str2 */
    for (j=0; str1[j]!='\0'; j++) {
        found = appearsIn(str1[j], str2);
        if (!found)
            return (0);
    }

    /* The symmetric check - if all characters from str2 are in str1 */
    for (j=0; str2[j]!='\0'; j++) {
        found = appearsIn(str2[j], str1);
        if (!found)
            return (0);
    }
    return (1);
}

```

18a/b. See the following program:

```

#define BOARD_SIZE 8 /* number of rows and columns on the chess board*/
#define EMPTY      0 /* representation of an empty board cell          */
#define KING       100 /* representation of a cell containing the king */
...                /* similar constants for other chess pieces      */

/* colors (as in GP142) */
#define BLACK      0
#define WHITE      1

typedef struct { /* definition of one chess board cell          */
    int pieceKind; /* kind of piece in this cell or EMPTY      */
    int pieceColor; /* color of this piece if cell is not EMPTY */
} cell;

/* The chess board */
cell board[BOARD_SIZE][BOARD_SIZE];

/* Returns true (1) if the square on the given row and column is */
/* threatened, and false (0) otherwise.                          */
int is_threatened(cell board[BOARD_SIZE][BOARD_SIZE],
                  int row, int col);

/* Search for the specified pieceKind with the specified pieceColor */
/* on the board. If the piece is found, set row and col to its      */
/* location and return true (1). If it is not found, return false (0)*/
/* and leave row and col unchanged.                                */
int find_piece(cell board[BOARD_SIZE][BOARD_SIZE],
               int pieceKind, int pieceColor, int *row, int *col)
{
    int k, j;
    for (k=0; k<BOARD_SIZE; k++)
        for (j=0; j<BOARD_SIZE; j++)
            if (board[k][j].pieceKind == pieceKind &&
                board[k][j].pieceColor == pieceColor)
            {
                *row = k;
                *col = j;
            }
}

```

```

        return(1);
    }
    return (0);
}

/* Returns true (1) if the king of the given color is in check */
/* (i.e. the square it currently occupies is threatened).      */
/* If the king is not threatened, return false (0).            */
int is_check(cell board[BOARD_SIZE][BOARD_SIZE], int kingColor)
{
    int row, col, resultFind;
    resultFind = find_piece(board, KING, kingColor, &row, &col);
    if (resultFind == FOUND)
        if (is_threatened(board, row, col))
            return (1);
        else
            return (0);
    else
    {
        printf("Incorrect position. One king is missing!");
        return (0);
    }
}

```