CS13002 Programming and Data Structures, Spring 2006

Class test 2

Total points: 30 | March 30, 2006 | Total time: 1 hour

Roll no: 05FB1331 **Name:** Foolan Barik **Section:** @

Write your answers in the question paper itself. You may use extra blank sheets for rough work, but your answers must fit in the respective spaces provided. Not all blanks carry equal marks. Evaluation will depend on the overall correctness of your solutions. Answer all questions.

1. Let $\omega = \sqrt[3]{2}$ be the real cube root of 2. Consider the set

$$A = \{a + b\omega + c\omega^2 \mid a, b, c \text{ are integers}\}$$

of real numbers. It turns out that the set A is closed under addition, subtraction and multiplication, i.e., the sum, difference and product of $a_1 + b_1\omega + c_1\omega^2, a_2 + b_2\omega + c_2\omega^2 \in A$ can be expressed in the form $a + b\omega + c\omega^2$. Clearly, $(a_1 + b_1\omega + c_1\omega^2) \pm (a_2 + b_2\omega + c_2\omega^2) = (a_1 \pm a_2) + (b_1 \pm b_2)\omega + (c_1 \pm c_2)\omega^2$.

For computing the product, first multiply $a_1 + b_1\omega + c_1\omega^2$ and $a_2 + b_2\omega + c_2\omega^2$ and obtain a polynomial in ω of degree 4. Then use the facts $\omega^3 = 2$ and $\omega^4 = 2\omega$ in order to reduce this polynomial of degree 4 back to a polynomial of degree 2. That is, we have:

$$\begin{aligned}
& (a_1 + b_1\omega + c_1\omega^2)(a_2 + b_2\omega + c_2\omega^2) \\
= \; & (a_1a_2) + (a_1b_2 + a_2b_1)\omega + (a_1c_2 + b_1b_2 + a_2c_1)\omega^2 + (b_1c_2 + b_2c_1)\omega^3 + (c_1c_2)\omega^4 \\
= \; & (a_1a_2) + (a_1b_2 + a_2b_1)\omega + (a_1c_2 + b_1b_2 + a_2c_1)\omega^2 + (b_1c_2 + b_2c_1) \times 2 + (c_1c_2) \times (2\omega) \\
= \; & (a_1a_2 + 2b_1c_2 + 2b_2c_1) + (a_1b_2 + a_2b_1 + 2c_1c_2)\omega + (a_1c_2 + b_1b_2 + a_2c_1)\omega^2.
\end{aligned}$$

Represent an element of A by a structure of three integers:

```
typedef struct {
   int a,b,c; /* Represents a + bω + cω² ∈ A */
} cubicNumber;
```

Complete the following function that takes two cubic numbers **x1**, **x2** as arguments and returns their product. **(8)**

```
cubicNumber cubicProd ( cubicNumber x1, cubicNumber x2 )
{
   cubicNumber x;   /* local variable */

   x.a = (x1.a) * (x2.a) + 2 * (x1.b) * (x2.c) + 2 * (x2.b) * (x1.c) ;
   x.b = (x1.a) * (x2.b) + (x2.a) * (x1.b) + 2 * (x1.c) * (x2.c) ;
   x.c = (x1.a) * (x2.c) + (x1.b) * (x2.b) + (x2.a) * (x1.c) ;

   return x;
}
```

2. A house has n rooms numbered $0, 1, \ldots, n-1$. Consider the $n \times n$ matrix M. The i, j-th entry of M is 1 if there is a door between room i and room j; it is 0 otherwise. Given two rooms u, v, the following function finds out whether there exists a way to go from room u to room v using the doors. The function works as follows. It maintains two arrays **`visited[]`** and **`toExplore[]`**. To start with, room u is marked as visited and is also inserted in the array **`toExplore[]`**. Then we enter a loop. We first check if there are some more rooms to explore. Let i be one such room. We find out all unvisited rooms j sharing doors with room i. We mark all these rooms j as visited and also insert them in the array **`toExplore[]`**. A good way to handle elements of the array **`toExplore[]`** is to insert elements at the end (of the current list of rooms to explore) and consider the rooms from beginning to end for further exploration. We maintain two indices **`start`** and **`end`**. The rooms yet to be explored lie in the indices **`start`**, …, **`end`** in **`toExplore[]`**. Complete the following function to solve this connectivity problem. **(22)**

```
int connected ( int M[MAXDIM][MAXDIM], int n, int u, int v )
{
   int *visited, *toExplore, i, j, start, end;

   /* Allocate memory for n integers to each of visited and toExplore */

   visited   = ______________(int *)malloc(n * sizeof(int))______________;

   toExplore = ______________(int *)malloc(n * sizeof(int))______________;
   for (i=0; i<n; ++i) visited[i] = 0; /* Initialize the array visited */

   visited[u] = ____________1____________ ; /* Mark room u as visited */
   /* Insert room u in the array toExplore */
   toExplore[0] = u; start = end = 0;
   /* As long as there are more rooms to explore */

   while ( ____________start <= end____________ ) {
      i = toExplore[start]; ++start;
      /* if i is the destination room v, return true */
      if ( i == v ) return 1;
      /* Check all rooms j sharing doors with room i */
      for (j=0; j<n; ++j) {
         /* if there is a door between i and j, and j is not visited */

         if ((_____M[i][j] == 1_____) && (_____visited[j] == 0_____)) {
            /* Mark j as visited */

            _______________________visited[j] = 1;_______________________
            /* Insert j in toExplore[] and adjust the insertion index */

            ____________________toExplore[++end] = j;____________________
         }
      }
   }
   /* Loop ends. Room v could not be reached from room u. */
   /* Free allocated memory */

   free(________visited________); free(________toExplore________);
   /* Return failure */

   ______________return 0;______________
}
```