CS13002 Programming and Data Structures, Spring 2006

Mid-semester examination

Total marks: 60 | February 2006 | Total time: 2 hours

Roll no: _____ **Section:** _____

Name: _____

- *This question paper consists of seven pages. Do not write anything on this page except your name, roll number and section.*
- *Answer all questions.*
- *Write your answers on the question paper itself. Your final answers must fit in the respective spaces provided. Strictly avoid untidiness or cancellations on the question-cum-answer paper.*
- *Do your rough work on the given answer-script or additional supplements. The rough work must be submitted, but will not be evaluated. Only answers in the question-cum-answer paper will be evaluated.*
- *Not all blanks carry equal marks for Questions 3, 4 and 5. Evaluation will depend on the overall correctness.*

(To be filled in by the examiners)

Question No	**1**	**2**	**3**	**4**	**5**	**Total**
Marks						

1. For each of the following parts, circle the letter corresponding to the correct answer. **(1×12)**

(a) Which of the following is not a legal floating-point constant in C?

(A) `+123.45` **(B)** `-12345.` **(C)** `+5.43e+21` **(D)** `-5.43e-2.1`

(b) What is the 8-bit 2's-complement representation of -33?

(A) 01011111 **(B)** 10100001 **(C)** 11011110 **(D)** 11011111

(c) Let the array `A` be initialized as:

```
char A[10] = "PDS2006";
```

After this initialization what is the character stored at `A[7]`?

(A) `'0'` **(B)** `'\0'` **(C)** `'6'` **(D)** Cannot be determined

(d) Assume that the integer variables `a`, `b` and `c` respectively store the values 7, 4 and -2 respectively. At that instance, the following statement is executed:

```
a = a - a % b * c;
```

What value is assigned to the variable `a`?

(A) 0 **(B)** 9 **(C)** 13 **(D)** 14

(e) Assume that `a`, `b` and `c` are integer variables holding the values $5, 3, 4$ respectively. Which of the following conditions is true in C?

(A) `(a<b)<c` **(B)** `(a<b)||(c<b)` **(C)** `(c<a)&&(c<b)` **(D)** `!(a+b>c)`

(f) Let `i`, `j` be integer variables. What is the value printed after the following loop?

```
for (i=j=0; i<10; ++i) {
    j = i + j;
    ++i;
}
printf("%d\n", j);
```

(A) 20 **(B)** 25 **(C)** 30 **(D)** 45

(g) Let `i`, `j`, `s` be integer variables. What is the value of `s` printed after the following outer loop?

```
s = i = 0;
while (i < 10) {
    j = 0;
    while (j < 5) {
        if (i != j) ++s;
        ++j;
    }
    ++i;
}
printf("%d\n", s);
```

(A) 15 **(B)** 40 **(C)** 45 **(D)** 50

(h) Let the function **f** be defined as follows:

```
int f ( int n )
{
    int t;
    t = 100000 * (n % 10) + (n / 10);
    return t / n;
}
```

What is the value returned by the call **f(142857)**?

(A) 0 **(B)** 3 **(C)** 5 **(D)** 10

(i) Let the function **g** be defined as follows:

```
int g ( int n )
{
    if (n < 2) return n;
    return g(n/2);
}
```

What is the value returned by the call **g(142857)**?

(A) 0 **(B)** 1 **(C)** 2 **(D)** 71428

(j) What is the value printed by the following program?

```
#include <stdio.h>
int h ( int a , int b ) { a = b - a; return b - a; }
int main ()
{
    int a = 9, b = 2;
    a = h(a,b);
    b = h(a,b);
    printf("%d\n", a);
}
```

(A) 0 **(B)** 2 **(C)** 7 **(D)** 9

(k) What is the output of the following program?

```
#include <stdio.h>
int n = 10;
void fgh ( int r ) { printf("%d,%d\n",n,r); }
int main ()
{
    int n = 20;
    fgh(n);
}
```

(A) 10,10 **(B)** 10,20 **(C)** 20,10 **(D)** 20,20

(l) How many times elements are swapped for bubble sorting an array of size four initially containing the integers $5, 3, 8, 2$?

(A) 3 **(B)** 4 **(C)** 5 **(D)** 6

2. **(a)** Consider the following program:

```
#include <stdio.h>

int main ()
{
   int a,b,c,d;
   int r,s,t;

   scanf("%d%d%d%d", &a, &b, &c, &d);
   if (a > b) r = a; else r = b;
   if (c > d) s = c; else s = d;
   if (r > s) t = r; else t = s;
   printf("%d\n", t);
}
```

Suppose that the user inputs the values $5, 3, 5, 9$ for **a**, **b**, **c** and **d** respectively. What value for **t** does the program print? **(2)**

_____

Describe in an English sentence the value, in terms of the input integers **a**, **b**, **c** and **d**, printed by the above program. **(4)**

_____

_____

_____

_____

(b) The following function expects a non-negative integer argument.

```
unsigned int doit ( unsigned int n )
{
   unsigned int i, s, t;

   s = 0; t = 1;
   for (i=0; i<n; ++i) {
      s = s + i + n;
      t = t * 2;
   }
   return s * t;
}
```

What value does **doit(4)** return? _____ **(2)**

Describe as a mathematical function of **n** the value returned by the function **doit(n)**. **(4)**

_____

_____

_____

_____

3. This exercise explains the *counting sort* algorithm. Suppose that the array **A** to be sorted is known to consist only of integers in a fixed range, say, between 0 and $B-1$. We maintain an array **C** of B counters each initialized to zero. We read the input array **A** element by element and increment the counter corresponding to each integer read. After the whole of **A** is read, the i-th counter stores the value how many times the integer i occurs in the input array. We then write down the elements $0, 0, \ldots, 0, 1, 1, \ldots, 1, \ldots, B-1, B-1, \ldots, B-1$, where each i in the range $0 \leqslant i \leqslant B-1$ is repeated as many times as the value accumulated in the i-th counter.

Complete the following program that implements the counting sort algorithm. **(12)**

```
#include <stdio.h>

#define B 1000
#define MAX_SIZE 100000

int main ()
{
   int A[MAX_SIZE];  /* The input array */

   /* The counter array. Supply the exact size needed. */

   int C[____________];

   int i, j, n;

   printf("Size of the input array A : "); scanf("%d",&n);

   /* Read the elements of the array A */
   printf("Enter %d integers each in the range 0 to %d\n", n, B-1);

   for (i=0; i<n; ++i) scanf("%d", ________________________ );

   /* Initialize each counter in the array C to zero */

   for (i=0; i< ____________; ++i) ____________________________________

   /* Read the input array element-by-element and
      increment the appropriate counters */

   for (i=0; i<____________; ++i) ++C[__________________________________];

   /* Write down the sorted list */
   for (i=0; i<B; ++i) {

      for (j=0; j<________________________; ++j) {

         printf("%d\n", ________________________ );
      }
   }
}
```

4. Recall that during each iteration of the binary search algorithm in a sorted array, we reduce the size of the search window to nearly half using one comparison only. The *ternary search* algorithm requires the input array to be sorted and during each iteration reduces the size of the search window to nearly one-third by making two comparisons. In each iteration of the ternary search algorithm one first computes two intermediate indices which are about one-third away from the left and right boundaries of the current search window. The iteration stops when the size of the search window reduces to 1.

Complete the following code for ternary search. **(12)**

```
#include <stdio.h>

int main ()
{
   int A[100000], n, key, L, R, M1, M2;

   printf("Size of the array :  "); scanf("%d", &n);
   for (i=0; i<n; ++i) scanf("%d",&A[i]);
   printf("Enter key to search :  "); scanf("%d", &key);

   /* Initialize the search window to the whole array */

   L = ________________________ ; R = ________________________;

   /* Loop as long as the search window has more than one elements */

   while ( ________________________ ) {
      /* Calculate the two intermediate indices */

      M1 = ____________________________________________________ ;

      M2 = ____________________________________________________ ;

      if (key <= A[M1]) {

         R = ____________________________________________________ ;
      } else if (key <= A[M2]) {

         L = ____________________________________________________ ;

         R = ____________________________________________________ ;
      } else {

         ____________________________________________________
      }
   }

   /* Make the final equality comparison */

   if ( ____________________________________ ) {
      printf("Key found in the array.\n");
   } else {
      printf("Key not found in the array.\n");
   }
}
```

5. In a country called Frobeniusia, coins of denominations u, v, w Frupees are available. Your problem is to find out in how many ways a sum of n Frupees can be exchanged with these coins. For example, if $u = 2$, $v = 3$ and $w = 5$, then $n = 8$ Frupees can be exchanged in exactly three different ways: $8 = 5 + 3 = 3 + 3 + 2 = 2 + 2 + 2 + 2$. In this case, $5 + 3$ is treated the same as $3 + 5$, $3 + 3 + 2$ to be the same as $3 + 2 + 3$ and as $2 + 3 + 3$.

In order to solve the problem, you are asked to write a recursive function. If $n < 0$ (an invalid value), the sum cannot be exchanged. If $n = 0$, there is a unique way of exchanging the sum, namely the empty change. If $n > 0$, we consider three cases: we choose a coin of one of the three denominations. We then recursively compute in how many ways $n - u$, $n - v$ and $n - w$ Frupees can be exchanged. We take the sum of the three values returned by the recursive calls.

There remains a small problem with the above algorithm, namely, it does not guarantee that the changes enumerated in the recursive way are distinct from one another. In order to ensure that, we select coins in non-increasing values of denomination. In the example above, once we select a 3 Frupee coin, we do not choose a 5 Frupee coin, but we are allowed to choose 3 and 2 Frupee coins. This choice is governed by passing an additional parameter m to the function, which indicates the smallest denomination chosen so far. Before the next recursive call, we avoid choosing a coin of denomination bigger than m.

Complete the following program to solve the coin change problem of Frobeniusia. **(12)**

```
#include <stdio.h>

int countChange ( int u , int v , int w , int n , int m )
{
    int cnt = 0;

    /* Termination cases */

    if ( ____________________ ) return ____________ ; /* Invalid sum */

    if ( ____________________ ) return ____________ ; /* The empty change */

    /* Here n > 0. Make three recursive calls conditionally. */

    if (u <= m) cnt += countChange(_________________________________________);

    if (v <= m) cnt += countChange(_________________________________________);

    if (w <= m) cnt += countChange(_________________________________________);

    return cnt;
}

int main ()
{
    int u, v, w, n, cnt;

    printf("Enter three distinct positive denominations : ");
    scanf("%d%d%d", &u, &v, &w);
    printf("Enter the sum to exchange : ");
    scanf("%d", &n);

    /* Make the outermost call of the function countChange(). */
    /* Supply a suitable value for the last argument. */

    cnt = countChange( u, v, w, n, ________________________ );
    printf("Number of ways of exchanging the given sum = %d\n", cnt);
}
```