National University of Singapore

School of Computing

FINAL EXAMINATION for Semester 2 AY2008/2009

CS1101C — Programming Methodology in C

25 April 2009 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This test paper contains 24 questions in two parts A and B, and comprises 14 printed pages, including this page.

2. The maximum possible mark is 40.

3. Answer *all* questions in the *space provided* in this booklet.

4. This is an *open book* examination.

5. Do not look at the questions until you are told to do so.

6. There is no negative marking, so please attempt every question.

Part A (Each question is worth 1 mark. Total: **20** marks)

In this part, you are required to write down the output of the given code fragment, or write down that there is an *error*.
An error includes compiler, runtime or infinite loop error.
Your answers are to be brief and to the point.
Do not include any commentary in the answers.

Assume that the relevant `#include` pre-processor statements have been included in the code fragment as appropriate.

1.
```
main()
{
    int i = 1, i2;
    unsigned int u, u2;
    double d = 2, d2;
    char c = '3', c2, c3;
    i2 = d / 3;
    d2 = (c - '0') / 2;
    c2 = 'a' + d - 0.0001;
    c3 = 'a' + d + 0.0001;
    printf("%d %.1f %c %c\n", i2, d2, c2, c3);
}
```

2.
```
#define square(x) x*x
main()
{
    int i;
    i = 64/square(4);
    printf("%d", i);
}
```

3.
```
main()
{
    int i = -1,j =- 1, k = 0, l = 2, m;
    m = i++&&j++&&k++||l++;
    printf("%d %d %d %d %d", i, j, k, l, m);
}
```

4.
```
main()
{
    int i = 1, j = 2;
    switch (i)
    {
       case 1: printf("good");
               break;
       case j: printf("bad");
               break;
    }
}
```

5.
```
main()
{
    int i, j = 0;
    for (i = 0; i < 999; i++) {
        switch (i) {
            case 0:
            case 1: if (i < 1) break; i = 555;
            case 2: if (i == 555) j = 666; break;
            default: j = 777;
        }
        if (i == 555) break;
    }
    printf("%d %d\n", i, j);
}
```

6.
```
main()
{
    int i = 0;
    do {
        if (i < 3) {
            i += 2;
            printf("%d\n", i);
            continue;
        }
        printf("%d\n", ++i);
    } while (i < 3);
}
```

7.
```
main()
{
    int i = 0;
```

```
    do {
        if (i < 3) {
            i += 2;
            printf("%d\n", i);
            break;
        }
        printf("%d\n", ++i);
    } while (i < 3);
}
```

8.
```
main()
{
    int i, j;
    for (i = 0; i < 3; i++)
        for (j = 0; j <= i; j++)
            printf("%d %d\n", i, j);
}
```

9.
```
int func(int n);
main()
{
    printf("%d\n", func(func(21)));
}
int func(int n)
{
    do {
        if ((n-- % 5) == 0) break;
    } while (1);
    return n;
}
```

10.
```
int main()
{
   func();
   func();
   return 0;
}

void func()
```

```
{
   int x = 0;
   static int y = 0;
   x++; y++;
   printf( "%d %d\n", x, y );
}
```

11. Describe the output of running the function call `fib(4)` on the following:

```
int fib(int n) {
    printf("%d ", n);
    return (n <= 1 ? n : fib(n - 1) + fib(n - 2));
}
```

12.
```
#define N 100
main()
{
    int i, j, count = 0;
    int num[N][N];
    int newnum[N*N];
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            num[i][j] = i*N + j;
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            newnum[count++] = num[i][j];
    printf("%d\n", newnum[67]);
    printf("%d\n", newnum[0]);
    printf("%d\n", newnum[876]);
}
```

13.
```
main()
{
    int i1, i2, *p1, *p2;
    i1 = 5;
    p1 = &i1;
    i2 = *p1 / 2 + 10;
    p2 = p1;
    printf("%d %d %d %d\n", i1, i2, *p1, *p2);
}
```

14.
```
main()
{
    int a = 2, *f1, *f2;
    f1 = f2 = &a;
    *f2 += *f2 += a += 2;
    printf("%d %d %d\n", a, *f1, *f2);
}
```

15.
```
main()
{
    char s[] = "man";
    int i;
    for(i = 0; s[i]; i++)
        printf("%c%c%c\n", s[i], *(s+i), *(i+s));
}
```

16.
```
#include <string.h>
main()
{
    char s1[] = "0xyzabc";
    char s2[] = "0xyz";
    char s3[] = "1xyz";
    s1[4] = 0;
    printf("%d %d %d\n", strcmp(s1, s2), strcmp(s2, s3), strcmp(s3, s2));
}
```

17.
```
#define N 9
void func(int a, int b[], int *c);

main()
{
    int a = 0, b[N] = {0};
    func(a, b, &b[1]);
    printf("%d %d %d %d\n", a, b[0], b[1], b[2]);
}

void func(int a, int b[], int *c)
{
    a = 1;
    b[1] = 2;
    *(c + 1) = 3;
}
```

18.
```
main( )
{
    int  a[] = {0, 1, 2, 3};
    int *p[] = {a, a+1, a+2, a+3};
    int **ptr =  p;
    ptr++;
    printf("%d%d%d\n", ptr-p, *ptr-a, **ptr);
    *ptr++;
    printf("%d%d%d\n", ptr-p, *ptr-a, **ptr);
    *++ptr;
    printf("%d%d%d\n", ptr-p, *ptr-a, **ptr);
}
```

19.
```
struct time {
   int *day;
   int *month;
   int *year;
};
struct time t1, *times;

main()
{
   int d = 25, m = 4, y = 2009;

   t1.day = &d;
   t1.month = &m;
   t1.year = &y;
   printf("%d %d %d\n", *t1.day, *t1.month, *t1.year );
   times = &t1;
   *(times->day) = 10;
   printf("%d %d %d\n", *t1.day, *t1.month, *t1.year );
}
```

20.
```
main()
{
    int p = 1;
    struct {
        int next;
        char name[10];
    } s[] = {{3, "cow"}, {2, "how"}, {0, "now"}, {-1, "brown"}};

    do {
        printf("%s ", s[p].name);
        p = s[p].next;
    } while (p >= 0);
    printf("\n");
}
```

Part B (Each question is worth 5 marks. Total: **20** marks)

In this part, the required code fragment is only a few lines.
Comments are not necessary; if you wish to provide comments
they must be restricted to a few words.

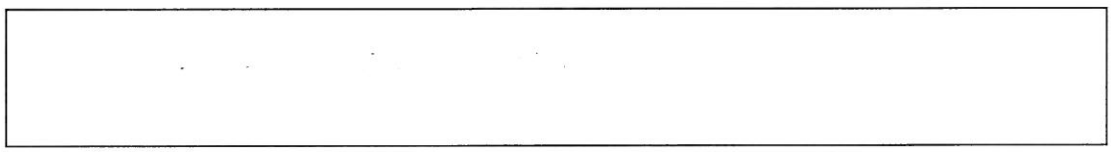
21. We wish to compute the number of months it takes to repay a loan. A fixed monthly payment and an interest rate of one percent *per month* is applicable. For example, given a loan of amount \$100 and a fixed monthly payment of \$40,

    After 1 month:
    you owe an additional $0.01 * \$100 = \$1$, make payment of \$40, so you now owe \$61.
    After 2 months:
    you owe an additional $0.01 * \$61 = \$0.61$, make payment of \$40, so you now owe \$21.61.
    After 3 months,
    you owe an additional $0.01 * \$21.61 = \$0.2161$, make payment of \$40, and now your loan is paid.
    (Actually, you have overpaid a bit, by \$18.1739, but we shall ignore this detail.)

    So the total time required was 3 months.

    Write two versions of a function that computes this loan duration. You may assume that the monthly payment exceeds the value of one month's interest on the loan.

    First write a ~~*recursive version*~~. Non-recursive version



Next write a ~~non~~*recursive* version.

22. A *magic square* is a square matrix where the sum of the elements in each row, column and diagonal is the same. An example where the number of rows is 3 and the sum in question is 15:

6	1	8
7	5	3
2	9	4

Write a function to determine if a given two dimensional array represents a magic square.

23. Use a structure to represent a *card* whose components are a *rank* from 1 to 13, and a *suit* from 1 to 4. Use an array to represent a *deck* of 52 cards. Write a code fragment to simulate a "shuffling" of the deck. Using the `rand` function, swap the first card with the card at a random position in the rest of the deck, that is, in the second through last positions. Then swap the second card with the card at a random position in the rest of the deck, that is, in the third through last positions. And so on.

    Write your code fragment here:

24. We wish to store information about students, modules and grades. Assume that there are S students and M modules in total.
    The following defines a student database: an array *students* of structures each of which represents an individual *student*. A student, in turn, is a structure whose components are a *matricnumber*, a collection of *modules*, and a *count* of the number of modules taken by the student in question. The type of each component is simply integer.

```
#define S ...
#define M ...

typdef struct student {
    int matricnumber;
    int modules[M];
    int count;
} students[S];
```

Define a module database. This should be an array *modules* where each element *module* is a structure with the following components: a *modulecode*, a collection of *students* together with a corresponding *grade*, and a *count* of the number of students having taken the module in question. The type of each component is simply integer. The value of the grade is a number from 0 to 5.

Make your declaration here:

Assume that the databases `students` and `modules` have already been populated with some data. For example, let $S = 2, M = 3$ and suppose the two students have matric numbers 111111 and 222222 and the module codes are 1101, 1102 and 1103. Suppose student 111111 took modules 1101, 1103 obtaining grades 3, 4 respectively. Suppose student 222222 took modules 1102, 1103 obtaining grades 1, 5 respectively. Then the average grade or CAP for the students are 3.5 and 3.0 respectively.

Write a code fragment to compute and print the CAP for every student.

14

end