NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1 AY2010/2011

CS1010 – PROGRAMMING METHODOLOGY

November 2010 Time allowed: 2 hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **EIGHT (8)** questions and comprises **TWELVE (12)** printed pages.
2. This is an **OPEN BOOK** examination.
3. Answer all questions.
4. Write your answers in the ANSWER SHEETS provided.
5. Fill in your Matriculation Number with a pen, clearly on every page of your ANSWER SHEETS.
6. You may use **2B pencil** to write your codes. Pen is preferred for other questions.
7. You must submit only the ANSWER SHEETS and no other document.

Multiple Choice Questions (MCQs): Q1.1 – Q1.4 **[8 marks]**

Each MCQ has one correct answer and is worth 2 marks. There is no penalty for wrong answer.

Q1.1 Assume that a C program was developed with 3 separate modules. The source files of the modules are **`main.c`**, **`search.c`** and **`storage.c`**. Which command should be executed next to produce an executable file from the created object files?

```
$ gcc –c main.c
$ gcc –c search.c
$ gcc –c storage.c
$ Which command should be used here?
```

A. `gcc main.c search.c storage.c –c main.o –lm`

B. `gcc –c main.o search.o storage.o –lm`

C. `gcc main search storage -lm`

D. `gcc -o main main.o search.o storage.o –lm`

E. `gcc -g a.out main.c search.c storage.c -lm`

Q1.2 Given the following function, what does **`calculate(5)`** compute?

```
int calculate(int n)
{
    if (n == 0)
        return 0;
    else
        return (2 * n + calculate(n-1));
}
```

A. 10

B. 14

C. 25

D. 30

E. 42

Q1.3 Refer to the function below.

```
void foo(char *fname)
{
   FILE *fp;
   int  c;
   int  count = 0;

   if ((fp = fopen(fname, "r")) == NULL)
      return;

   while (((c = fgetc(fp)) != EOF) && (count < 3))
   {
      if (c == '\n')
         count++;
      putchar(c);
   }
   fclose(fp);
}
```

If "dat.txt" is a text file containing 100 characters, which of the following statements is the most appropriate about the output of **`foo("dat.txt")`**?

A. The output produced by **`foo`** contains two characters.

B. The output produced by **`foo`** contains three characters.

C. The output produced by **`foo`** contains two newline characters.

D. The output produced by **`foo`** contains three newline characters.

E. None of the above.

Q1.4 Suppose you are doing your C programming on sunfire. Your program is buggy and when you execute it, it goes into an infinite loop. What should you press to break out of your program properly, thus terminating it?

A. ctrl-c

B. ctrl-x

C. ctrl-z

D. esc

E. ZZ

Q2. For each of the following programs, write out its output. **[15 marks]**

(a) [2 marks]

```
#include <stdio.h>

typedef struct
{
    int i, a[4];
} mystruct_t;

int main(void)
{
    mystruct_t s, t;
    s.i = 5;
    s.a[3] = 10;
    t = s;
    printf("%d %d\n", t.i, t.a[3]);
    return 0;
}
```

(b) [4 marks]

```
#include <stdio.h>

int functionXYZ(char *, char);

int main(void)
{
    char s[] = "abbacadaba";
    printf("%d\n", functionXYZ(s, 'b'));
    return 0;
}

int functionXYZ(char *str, char ch)
{
    int i=0, j=0;
    while (str[i])
        if (str[i++] == ch)
           j++;
    return j;
}
```

Q2. (cont.)

(c) [4 marks]

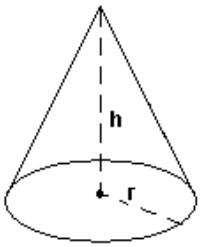
```
#include <stdio.h>

void swap(int []);

int main(void)
{
    int a[3] = {1, 2, 3};
    swap(a);
    printf("%d %d %d\n", a[0], a[1], a[2]);
    swap(&a[1]);
    printf("%d %d %d\n", a[0], a[1], a[2]);
    return 0;
}

void swap(int x[])
{
    int temp = x[0];
    x[0] = x[1];
    x[1] = temp;
}
```

Q2. (cont.)

(d) [5 marks]

```
#include <stdio.h>
#define N 5

void blah(int, int, int *, int []);

int main(void)
{
    int i, j = 1;
    int x[N] = {10, 2, 7, 9};

    blah(j, x[2], &x[3], x);

    for (i=0; i < N; i++)
        printf("%d ", x[i]);
    printf("\n");

    return 0;
}

void blah(int m, int p, int *q, int r[])
{
    int i;

    *q = *q + 3;
    p++;
    r[r[m]]++;
    for (i=0; i<N; i++)
        r[i] = r[i] + 2;
}
```

Q3. The program below computes the volume of a cone given its radius and height. The formula for volume of a cone is given as follows:



$$V = \frac{1}{3}\pi r^2 h$$

The program is messy (for example, incorrect use of parameters and variables). Though it compiles and runs, it gives completely incorrect result.

Correct the program so that it gives the right result. Corrections are restricted to crossing out, replacing, or adding parameters/variables/constants. **[5 marks]**

```
#include <stdio.h>
#define PI 3.14

float compute_volume(float, float, float, float);

int main(void)
{
    float radius, height, ans, r2;

    printf("Radius: ");
    scanf("%f", &radius);
    printf("Height: ");
    scanf("%f", &height);
    ans = compute_volume(radius, height, ans, r2);
    printf("Volume = %f\n", ans);

    return 0;
}

float compute_volume(float r, float h, float ans, float r2)
{
    float Ans;

    r2 = r*r;
    Ans = (1/3) * PI * r2 * h;

    return ans;
}
```

Q4. Recursion **[6 marks]**

Write a recursive function **`int largest_digit_pairs(int n)`** to determine the largest pair of digits of a positive integer **`n`** starting from the right to the left.

For example, if **`n`** is 5064321, then the pairs are 21, 43, 6 and 5, and hence the answer is 43.

No mark will be given for a non-recursive solution.

Q5. String **[6 marks]**

Write a function **`void convert_string(char *str, char *dest)`** that converts **`str`** into **`dest`** by adding an asterisk between each letter in **`str`**. Any blank space in **`str`** is also replaced by an asterisk.

You may assume that there is one blank space between two words, and only letters and spaces appear in **`str`**. You may also assume that **`dest`** has sufficient space to hold the lengthened string.

For example, if **`str`** is

```
The quick brown fox
```

then **`dest`** will be

```
T*h*e*q*u*i*c*k*b*r*o*w*n*f*o*x
```

Q6. Two-dimensional Array **[8 marks]**

Consider a two-dimensional $R \times C$ integer array M where each element in M is a non-negative integer between 0 and 9 inclusive. We say that M contains a pair with value v if there exists two consecutive elements within the same row or column in M that have the value v.

For example, consider the following array M, where $R = 4$ and $C = 4$.

	0	**1**	**2**	**3**
0	8	1	1	2
1	5	5	1	0
2	4	2	1	6
3	1	8	8	2

M contains a total of five pairs:

- Pair 1: M[0][1] and M[0][2] with a value of 1
- Pair 2: M[0][2] and M[1][2] with a value of 1
- Pair 3: M[1][0] and M[1][1] with a value of 5
- Pair 4: M[1][2] and M[2][2] with a value of 1
- Pair 5: M[3][1] and M[3][2] with a value of 8

You may assume that two constants R and C have been defined for the size of the two-dimensional array.

Write a function:

```
int max_pairs(int mat[][C])
```

that returns the maximum number of pairs of the same value contained in an input $R \times C$ integer array *mat*. In the above example, **`max_pairs(M)`** returns 3 corresponding to the number of pairs with a value of 1.

(Hint: Each array element is a non-negative integer value between 0 and 9 inclusive.)

Q7. Certificate of Entitlement **[12 marks]**

The Certificate of Entitlement (COE) is designed to limit the number of vehicles on the roads by requiring potential car owners to first obtain the right to buy a vehicle. Each month, the number of available COEs is made known and people who wish to buy a vehicle will submit their COE bids to a bidding system.

Suppose the number of available COEs for a particular month is *N*. At the end of the bidding cycle, we obtain the N^{th} highest bid as the COE Candidate Price. For example, if there are 4 COEs available for the month of March and the bids received are {1, 100, 50, 2, 8, 10000, 1000, 2, 1000, 10010}, then the COE Candidate Price is the 4^{th} highest bid, which is $1000. This is also the COE Final Price.

In the event that the number of bids that are greater than or equal to COE Candidate Price is more than *N*, then the COE Final Price is set to the next highest bid that is greater than COE Candidate Price.

For example, if there are 4 COEs available for the month of April and the bids received are {1, 100, 2000, 2, 8, 10000, 1000, 2, 1000, 10010}, then the COE Candidate Price is $1000. However, there are 5 bids that are more than or equal to $1000. Hence, the COE Final Price for April is the next highest bid (you may assume that you can always find one) that is more than $1000, which is $2000.

Write a complete C program that takes as input the number of available COEs for the month and the list of bid amounts (terminated by a zero bid amount), and outputs the COE Final Price for that month. Bid amounts are integers. You may assume that there are no more than 5000 bids received per month.

Two sample runs are shown below.

```
Number of available COEs: 4
Enter bids: 1 100 50 2 8 10000 1000 2 1000 10010 0
COE Final Price this month is $1000
```

```
Number of available COEs: 4
Enter bids: 1 100 2000 2 8 10000 1000 2 1000 10010 0
COE Final Price this month is $2000
```

Q8. Cards [20 marks]

Write a C program to model a deck of playing cards. A deck consists of 52 cards, each card containing a **rank** and a **suit**. The ranks are (from lowest to highest) 2, …, 9, 10, Jack, Queen, King and Ace, represented by the characters '2', …, '9', 'T', 'J', 'Q', 'K' and 'A' respectively. The suits are (from lowest to highest) Clubs, Diamonds, Hearts and Spades, represented by the characters 'C', 'D', 'H' and 'S' respectively.

(a) Define a structure called **`card_t`** to represent a playing card, and declare an array of cards, called **`deck`**, to represent the deck of cards.

(b) Write a function called **`initDeck`** to initialize the deck, i.e. create the deck of cards. The deck should be created starting with all cards of the lowest suit, and for cards of the same suit, in increasing order of their ranks. Hence, the first card is the 2 of Clubs and the last card is the Ace of Spades.

After the **`initDeck`** function is called, the function **`printDeck`** (which is given) is called and it gives the following output with each line showing 13 cards:

```
C2 C3 C4 C5 C6 C7 C8 C9 CT CJ CQ CK CA
D2 D3 D4 D5 D6 D7 D8 D9 DT DJ DQ DK DA
H2 H3 H4 H5 H6 H7 H8 H9 HT HJ HQ HK HA
S2 S3 S4 S5 S6 S7 S8 S9 ST SJ SQ SK SA
```

(c) Write a function called **`shuffleDeck`** to shuffle the deck by writing a loop with 52 iterations. At iteration *i* we generate a random integer *r* in the range [0, 51] and swap deck[*r*] with deck[*i*].

For example, after shuffling the deck might look something like this (i.e. the cards are now randomly ordered):

```
D9 C2 H9 S6 C8 D3 H2 DJ S3 D7 S8 H3 HQ
SJ CA DQ C4 D4 H4 HT CJ C3 HJ D8 H7 SK
C7 S4 DT D2 SQ H6 DA H8 ST C5 S7 S9 DK
D6 S2 CT HA C6 SA D5 CK C9 S5 CQ H5 HK
```

(d) The first 13 cards in the shuffled deck obtained in (c) will be given to a player. Write a function called **`computePoints`** to determine the total point of these 13 cards. This is determined by adding up the points for each suit. A suit has 3 points if it has no cards; 2 points if it has only one card; and 1 point if it has two cards. Otherwise, the suit has 0 point.

For example, the first 13 cards in the shuffled deck in (c) are as follows:

```
D9 C2 H9 S6 C8 D3 H2 DJ S3 D7 S8 H3 HQ
```

Here, we have 4 diamonds, 4 hearts and 3 spades, and they do not contribute any point. The 2 clubs, however, contribute 1 point. Hence this player has 1 point.

As another example, suppose the first 13 cards of another shuffled deck are as follows:

```
C7 CA DQ C4 C2 H4 HT CJ C3 HJ C8 H7 HK
```

We have 7 clubs, 1 diamond, 5 hearts, and no spade. The 1 diamond contributes 2 points and the absence of spades contributes 3 points, hence a total of 5 points.

The function prototypes are shown below.

```
void initDeck(card_t []);
void printDeck(card_t []); // code given
void shuffleDeck(card_t []);
int computePoints(card_t []);
```

The template for the program is already provided in the Answer Sheets. Fill in the required parts as requested.

=== END OF PAPER ===