ANSWER SHEETS

INSTRUCTIONS TO STUDENTS

1. This document consists of **SEVEN (7)** printed pages.
2. Fill in your Student Number clearly below. Do NOT write your name.

STUDENT NO.: | A | | | | | | | | |

(Write your Student Number legibly with a pen.)

For examiners' use only		
Question	***Max***	***Marks***
Q1-6	18	
Q7	6	
Q8	6	
Q9	5	
Q10	7	
Q11	26	
Q12	12	
Total	**80**	

MCQs [18 marks]

Q1	Q2	Q3	Q4	Q5	Q6
D	C	A	B	E	D

Q7. [6 marks]

(a) Pre-cond: $n > 2$

(b)

```
float f(int arr[], int n) {
   float average, min_average;
   int i;

   min_average = (arr[0] + arr[1] + arr[2])/3.0;
   for (i=2; i<n-1; i++) {
      if ((arr[i-1] < arr[i]) && (arr[i] < arr[i+1])) {
         average = (arr[i-1] + arr[i] + arr[i+1])/3.0;
         if (average < min_average) {
            min_average = average;
         }
      }
   }
   return min_average;
}
```

Note:

- The original code uses 'continue' indiscriminately. You can see that the code can be rewritten by removing all the 'continue' statements.
- There is redundant initialization of average and multiple lines of code to compute average which could be written on a single line.
- The empty 'else' statement is redundant.
- The original code computes the minimum average and also prints it. Converting the function into one that returns the computed value makes it cohesive.

Q8. [6 marks]

```
a = 55   , c = 15   , e = 0
*b = 55  , *d = 55  , *f = 0
```

Q9. [5 marks]

```
int IsGP(int a, int b, int c) {
    return ((a*b == c*c) || (a*c == b*b) ||
            (c*b == a*a));
}
```

Alternative answers possible. This answer avoids the use of division and math functions such as pow() which may give inaccurate result.

Q10. [Total: 7 marks]

(a) Representing flight cost data [1 mark]

Use a matrix (2D array) flights[100][100]
where flights[*src*][*dest*] contains the cost of the direct flight from city *src* to city *dest*.

(b) [6 marks]

```
// flights: the 2D array defined in part (a)
// src: source city; dst: destination city
Algorithm best_route(flights[][100], src, dst)

    min_cost ← MAX_INT (or some other appropriate initialisation)
    for city from 0 to 99 {
        if (city ≠src and city ≠ dst) {
            cost = flights[src, city] + flights[city, dst]
            if (cost < min_cost)
                min_cost = cost;
                stop_over = city;
        }
    }
    Return min_cost and stop_over
```

Q11. [Total: 26 marks]

(a) Define a constant and declare a suitable array of structures. [4 marks]

```
#define MAX_BOOKS 800
book_t book_list[MAX_BOOKS];
```

(b) read_file() function [12 marks]

```
/* Read the book information from the given file */
int read_file(char filename[], book_t booklist[]){
  int rec_read = 0, len;
  char ch;

  FILE *fptr;
  if ((fptr = fopen(filename, "r")) == NULL) {
    printf("Cannot open input file.\n");
    exit(1);
  }

  while (fgets(booklist[rec_read].title,MAX_STRING_LEN, fptr)
          != NULL) {
    len = strlen(booklist[rec_read].title);
    if (booklist[rec_read].title[len-1] == '\n')
      booklist[rec_read].title[len-1] = '\0';

    fgets(booklist[rec_read].author, MAX_STRING_LEN, fptr);
    len = strlen(booklist[rec_read].author);
    if (booklist[rec_read].author[len-1] == '\n')
      booklist[rec_read].author[len-1] = '\0';

    fgets(booklist[rec_read].isbn, ISBN_LEN, fptr);
    len = strlen(booklist[rec_read].isbn);
    if (booklist[rec_read].isbn[len-1] == '\n')
      booklist[rec_read].isbn[len-1] = '\0';

    fscanf(fptr, "%f", &booklist[rec_read].price);
    fscanf(fptr, "%d", &booklist[rec_read].qty);

    fscanf(fptr, "%c", &ch); // to skip the newline after qty

    rec_read++;
  }
  fclose(fptr);
  return rec_read;
}
```

Q11

(b) (continue here if you need to.)

Q11

(c) print_books() function [5 marks]

```
void print_books(book_t book_list[], int size){
   int i;

   for (i=0; i<size; i++)
      printf("%s, by %s, ISBN %s, SGD%.2f, Qty %d.\n\n",
             book_list[i].title,
             book_list[i].author,
             book_list[i].isbn,
             book_list[i].price,
             book_list[i].qty);
}
```

(d) main() function [5 marks]

```
int main(void){
   book_t book_list[MAX_BOOKS];
   int book_count;

   book_count = read_file("books.dat", book_list);
   print_books(book_list, book_count);

   return 0;
}
```

Q12. Alternating borders of 1 [12 marks]

Alternative answers possible. Here are just two.

```
void borderMatrix(int mtx[][MAX_COL],
                  int size, int start, int stop) {
   int row, col, n;

   if (start <= stop) {
      for (row=start; row<=stop; row++) {
         for (col=start; col<=stop; col++) {
            if (row == start || row == stop)
               mtx[row][col] = 1;
            else
               if (col == start || col == stop)
                  mtx[row][col] = 1;
         }
      }
      borderMatrix(mtx, size, start+2, stop-2);
   }
}
```

```
void borderMatrix(int mtx[][MAX_COL],
                  int size, int start, int stop) {
   int i;

   if (start <= stop) {
      for (i = start; i <= stop; i++) {
         mtx[start][i] = 1;
         mtx[i][start] = 1;
         mtx[stop][i] = 1;
         mtx[i][stop] = 1;

      }
      borderMatrix(mtx, size, start+2, stop-2);
   }
}
```

— END OF PAPER —

Comments

1. Multiple Choice Questions (MCQs)

	Q1	Q2	Q3	Q4	Q5	Q6
%students who chose the correct answer	D (86.0%) Easiest	C (52.1%)	A (73.0%)	B (23.1%) Hardest	E (80.1%)	D (53.1%)
%students who chose the most popular wrong answer	A (7.5%)	E (27.7%)	E (16.0%)	A (29.6%)	D (16.3%)	A (26.7%)

Question 1 is a give-away. Most students got it right.

Question 2 is a tracing question that that requires care. The following shows the content of array *arr* after every iteration of the outer loop:

Original array:	2	1	4	0	3
After 1st iteration:	2	4	5	2	3
After 2nd iteration:	4	4	5	6	3
After 3rd iteration:	5	7	5	6	3
After 4th iteration:	**5**	**7**	**9**	**6**	**3**

Question 3 requires knowledge of operator precedence. Given a=5, b=1, c=3, the assignment statement

```
a += --b || c++ * b;
```

is equivalent to the following:

```
a += ( (--b) || (c++ * b) );
```

We evaluate the expression on the right as follows:

- The value of **b** is decremented to 0; as 0 is false, we continue to evaluate **c++ * b**
- The current value of **c** (3) is used before it is incremented to 4. Hence **c++ * b** evaluates to 0 (false).
- Since the logical OR operation evaluates to false, 0 is added to **a**, hence **a** remains as 5.
- Output is: **a = 5; b = 0; c = 4**

How **question 4** ended up as the hardest MCQ is surprising. Since **str1** and **str2** are both referring to the same string literal "Hello", they are pointing to the same address, hence the first printf() statement prints "equal". The function **strcmp(str1, str2)** returns 0, and since 0 means false, the second printf() statement prints "not equal".

For **question 5**, the table shows the values of $f(n)$ for the first few values of n:

n	1	2	3	4	5
f(n)	3	6	10	15	21

It can be seen that $f(n) = (n+1)(n+2)/2$.

Question 6 is a very straight-forward question. Since a structure is passed by value to the function, the original value in **`s1`** is not changed by the function **`doSomething()`**. However, only slightly more than half the class got this right.

2. **Question 7:** Rewriting a program (Graded by Zhuohong)

   This is to test students' knowledge on programming style. The 'continue' statements are superfluous. The empty 'else' statement is redundant. All these can be removed without affecting the correctness of the program, and it makes the program shorter and easier to read. The multiple assignment statements for the computation of average can be combined into one.

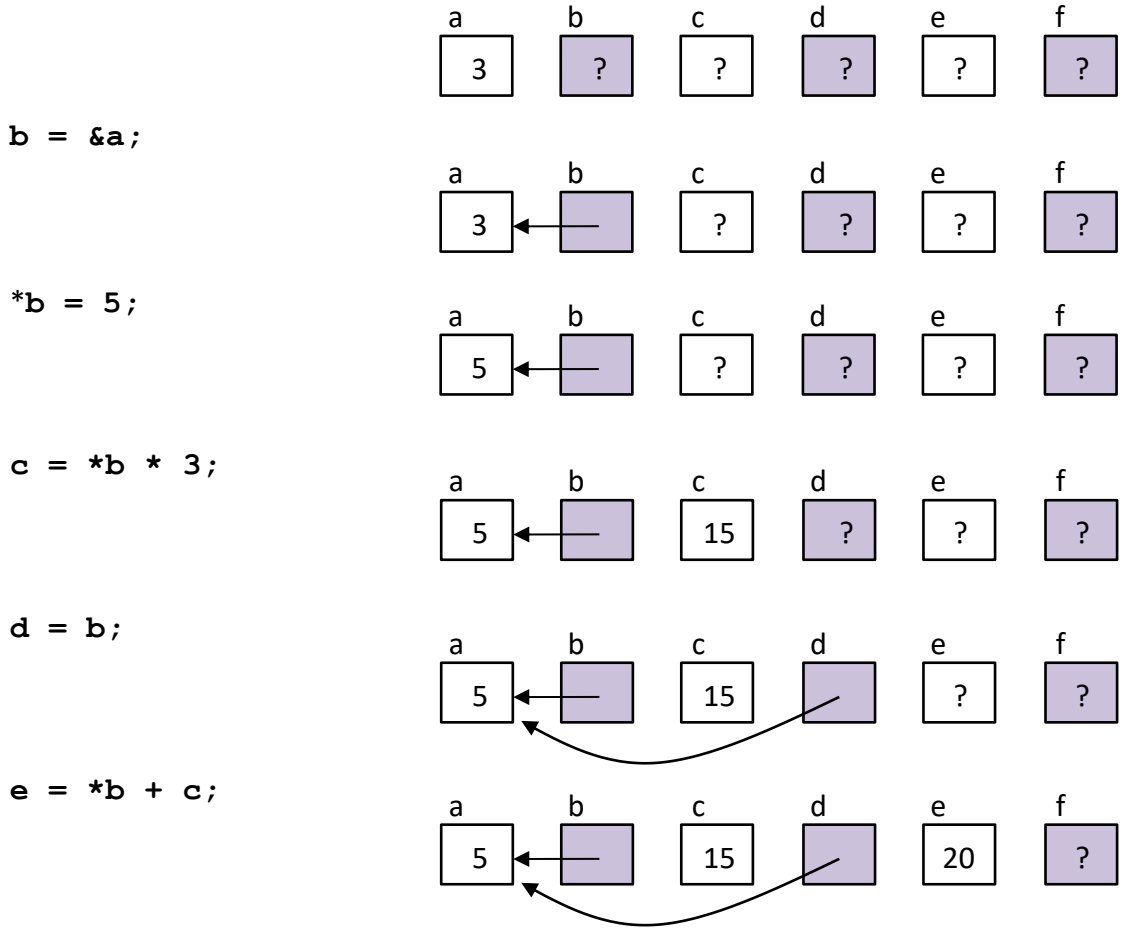
   What escaped many students is that the function should be converted to one that returns the computed min_average, instead of printing it. This is to improve function cohesion.

   Many students also didn't manage to write the correct pre-condition.

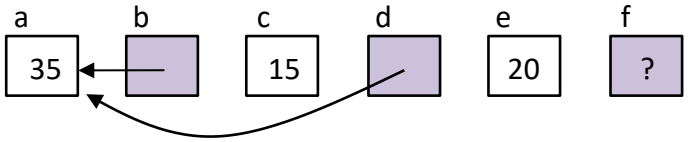
   The average mark for this question is 2.91/6, or 48.5%.

3. **Question 8:** Tracing a program on pointers (Graded by Aaron)

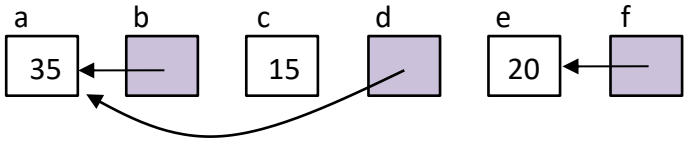
   The following shows the snapshots of the variables at each step.



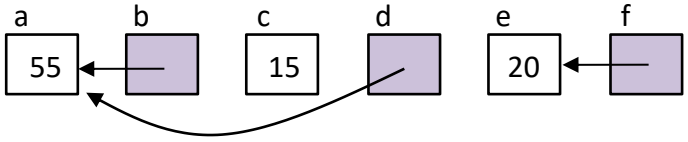
```
*d = c + e;
```



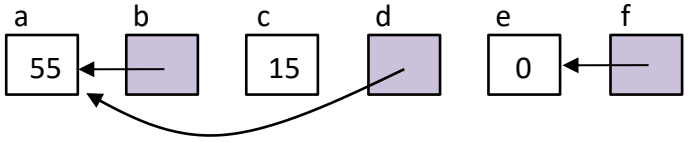

```
f = &e;
```




```
a = *f + *b;
```




```
*f = *d - *b;
```



The output is therefore:

```
a = 55, c = 15, e = 0
*b = 55, *d = 55, *f = 0
```

This question has the highest average mark of 4.37/6, or 72.9%.

4. **Question 9:** Geometric Progression (Graded by Aaron)

The most common mistake made by students is forgetting that / performs integer division with truncation if both operands are integers. Many students wrote such a code:

```
return ( (c/b == b/a) || (b/a == a/c) || … );
```

If a = 3, b = 5, c = 7, c/b and b/a would both be 1, and the above will return 1 (true), which is wrong.

Another common mistake is this:

```
return ( (b%a == c%a) || (c%b == a%b) || … );
```

Here, it returns the correct value for 2, 4, 8. However, for 2, 4, 16 which is not a geometric progression, it returns 1 (true) because 4%2 and 16%2 are both zero.

Some students wrote very complicated expression or a number of statements, ignoring the instruction that only a single statement would fetch full credit.

A number of such mistakes could easily be avoided had the students tested the function with a few simple examples.

The average mark for this question is 1.87/5, or 35.6%, making it the worst-attempted question. ☹

5. **Question 10:** Flights (Graded by Aaron)

   The easiest way to represent the flight cost data is to use a 100×100 two-dimensional array, flights[100][100], where flights[*src*][*dest*] represents the cost of the flight from source city *src* to destination city *dest*. This representation allows you to give a very simple algorithm for part (b).

   Alternatively, you could use an array of structures where each structure consists of the source city, destination city and the cost of flight from source to destination. However, using such a representation would require a slightly longer algorithm for part (b).

   Some students who used the second representation declared the array with 100 elements, which is not enough. Since there is a direct flight between any pair of cities and there are 100 cities and two directions, you need an array of $n \times (n - 1)$ elements. (This is equivalent to the 100×100 array without the diagonal elements.)

   Some students wrote very scanty answer for part (a), such as "an array" or "a 2D array". I had to look for more information about this array in the answer for part (b), failing which I would not award mark.

   Using the first representation (2D array) in part (b), the algorithm for part (b) is a straight-forward for loop.

   Some students wrote very sketchy algorithm for part (b) which is more like English statements rather than pseudocode.

   The average mark for this question is 2.94/7, or 42.0%.

6. **Question 11:** Book records (Graded by Gary)

   This is a question on array of structures and file processing.

   Some common mistakes encountered for part (b):

   - Using fscanf() to read in title, author. This is wrong as they will only read till the first space character.
   - Forgetting to remove the newline character \n and replacing it with null character \0 after fgets().
   - Need to skip the newline character \n at the end of each record.
   - Not using fclose().

   For part (c), the common mistake is forgetting to add in the newline character \n for the blank line between prints.

   The average mark for this question is 17.02/26, or 65.4%.

7. **Question 12:** Alternating borders of 1 (Graded by Zhuohong)

   This is a question on recursion. No mark is awarded if recursion is not used, as instructed in the question.

   The graded points are:

   1. Correct base case: 3 marks
   2. Correct steps to fill in the 1 and 0: 3 marks
   3. Correct recursion call: 3 marks
   4. Basic marks: 3 marks

   For this question, usually when a student knows how to do it, the answer is quite simple and straight forward and there is not much detail to handle. But when a student doesn't know how to do it (or maybe due to time limit since this is the last question), he/she can hardly make any attempt. That's why there are about same number of students (about 100 each) who scored 0 mark and full mark for this question.

   The average mark for this question is 5.79/12, or 48.3%.

Prepared by Aaron

5 December 2017