CONFIDENTIAL

UNIVERSITI TEKNOLOGI MALAYSIA

TEST 1
SEMESTER 1 2024/2025

SUBJECT CODE	**:**	**SECJ2013**
SUBJECT NAME	**:**	**DATA STRUCTURE AND ALGORITHMS**
YEAR/PROGRAM	**:**	**2 SECB / SECJ / SECP / SECR / SECV**
DURATION	**:**	**2 HOURS**
DATE/TIME	**:**	**20 Nov 2024 / 8:00 PM – 10:00 PM (MYT)**
VENUE	**:**	**N28 (BK1 – BK7)**

INSTRUCTIONS:

THIS QUESTION BOOK CONSISTS OF 2 SECTIONS:

SECTION A: 20 Objective Questions 20 Marks

SECTION B: 5 Structured Questions 80 Marks

ANSWER ALL QUESTIONS IN THE ANSWER BOOKLET PROVIDED
THE QUESTION BOOKLET MUST BE RETURNED WITH THE ANSWER BOOKLET

(Write your lecturer's name and section in the question, and answer booklets)

Name	
IC	
Year/Program	
Section	
Lecturer Name	

This question book consists of **15** printed pages excluding this page.

SECTION A: OBJECTIVE QUESTIONS (20 Marks)

Section A consists of 20 objective questions. Choose the correct answer, and write your answer in the answer booklet. Each question carries 1 mark.

1. Siti Maria and her team are collaborating with Universiti Teknologi Malaysia (UTM) to examine the existing student information system. They aim to understand the system's strengths and weaknesses and gather detailed requirements from UTM staff and students. This phase of the software development lifecycle is known as _____.

   A. Design
   B. Testing
   C. Analysis
   D. Maintenance

2. Which of the following best describes a queue?

   i. Elements are added at the back and removed from the front.
   ii. Allows removal of elements from both ends.
   iii. Operates on the LIFO (Last In First Out) principle.
   iv. Operates on the FIFO (First In First Out) principle.

   A. i and iv
   B. ii and iii.
   C. ii, iii and iv
   D. All the above

3. Given an array: [10, 2, 5, 7, 4] to be managed by using a stack. What will the contents of the array be after removing two items?

   A. [10, 5, 4]
   B. [2, 5, 7]
   C. [5, 7, 4]
   D. [10, 2, 5]

4. Which of the following is NOT the benefit of data abstraction by using object-oriented approach?

   A. Ease of maintenance and modification of code
   B. Increase the speed of computer to run the complex algorithm
   C. Reducing code complexity by hiding unnecessary details
   D. Reuse of data model in different type of algorithm/application

5. The `pList` array of a class `Person` has been declared as follows:

```
Person pList[5];
```

Which of the following is the correct syntax to call the `getName` method of the class by using the declared array?

A. `pList[0]->getName();`
B. `pList[0].getName();`
C. `pList.[0]->getName();`
D. `pList->[0].getName();`

6. Which of the following is NOT the benefit of using binary file format to save data?

A. Binary files are human-readable and easy to edit with a text editor.
B. Binary files allow faster read and write operations compared to text files.
C. Binary files save space by storing data in a compact format.
D. Binary files maintain data accuracy without converting data types.

7. In a recursive function, what is the role of the base/terminal case?

A. Divide the problem into smaller sub problems
B. Serves as condition to stop the recursive call
C. Calculate the final result after all recursion calls are completed
D. To continue making recursive call

8. Which of the following is true about the memory usage of recursive functions?

A. Use constant memory regardless of input size
B. Use less memory than standard iterative solutions (for and while)
C. Use more memory because each call adds a new layer to the call stack
D. No effects on computer memory usage

9. How many times is the `count` function called when the parameter `n` is passed with a value of three (3)?

```
void count(int n) {
  if (n > 0) count(n-1);
  cout << n << " ";
}
```

A. 1
B. 2
C. 3
D. 4

10. Based on the question 9 above, what is the output produced by the `count` function?

A. 0 1 2 3
B. 1 2 3
C. 3 2 1 0
D. 3 2 1

11. A recursive program is to be implemented as follows:

```
01  void iterate(int i) {
02    _________
03  }
04  int main() {
05    iterate(4);
06  }
```

Referring to the instruction to be filled in line 02 above, which of the following instruction will cause infinite call to the `iterate` function?

```
A. if (i > 3) iterate(i - 1);
B. if (i > 3) iterate(i + 1);
C. if (i < 3) iterate(i - 1);
D. if (i < 3) iterate(i + 1);
```

12. Which of the following Big-O notations represents the fastest-growing time complexity?

A. $O(n^2)$
B. $O(n \log_2 n)$
C. $O(2^n)$
D. $O(\log_2 n)$

13. Which of the following best describes the time complexity of an algorithm with three nested loops, each iterating n times?

    A. O(3n)
    B. $O(n^3)$
    C. $O(3^n)$
    D. $O(3 \log_3 n)$

14. What is the correct expression to represent the total number of steps in the following loop?

```
for (int i = 3; i < n + 2; i++)
```

    A. n - 3 + 2
    B. n + 3 - 2
    C. n - 3
    D. n + 2

15. Which of the following conditions helps to reduce the number of passes required in bubble sort?

    A. Check if the largest element is at the end in each pass
    B. Sort only half the array in each pass Answer 2
    C. Add a flag to detect if any swap occurred in a pass
    D. Swap the first and last elements in each pass

16. Which of the following sorting algorithms repeatedly compares adjacent elements and swaps them if they are in the wrong order?

    A. Bubble sort
    B. Insertion sort
    C. Quick sort
    D. Selection sort

17. Consider the array: {7, 4, 8, 2, 9}. What will the array look like after 3 passes of insertion sort in descending order?

    A. {8, 7, 4, 2, 9}
    B. {8, 7, 4, 9, 2}
    C. {8, 7, 4, 2, 9}
    D. {9, 8, 7, 4, 2}

18. Consider the array: {2, 5, 6, 4, 7}. How many swap operations are performed after 4 passes of selection sort in ascending order?

    A. 1
    B. 2
    C. 3
    D. 4

19. What is the time complexity of the Merge Sort algorithm when sorting an already sorted array?

    A. $O(n^2)$
    B. $O(n \log_2 n)$
    C. $O(n)$
    D. $O(\log_2 n)$

20. Which of the following is/are true about the quick-sort algorithm?

    i. It has a quadratic time complexity.
    ii. It is suitable for large datasets.
    iii. It is an in-place sorting algorithm.
    iv. In the best case, it divides the array into equal halves.

    A. i, ii, and iv
    B. ii, iii, and iv
    C. ii and iv
    D. iii only

SECTION B: STRUCTURED QUESTIONS (80 Marks)

Section B consists of 5 structured questions. Answer all questions and write your answer in the answer booklet.

Question 1 (10 Marks)

a) Write a `Product` class with three (3) member data (`name`, `price`, `unit`), constructors, mutators, accessors in C++ so it can be used and produced an output as follows:

[5 marks]

Example use of the `Product` class:

```
01 Product p1("Table", 25.50, 2);
02
03 Product p2("Chair");
04 p2.setPriceUnit(13.30, 4);
05
06 cout << p1.getName() << " " << p1.getTotalPrice() << "\n";
07 cout << p2.getName() << " " << p2.getTotalPrice() << "\n";
```

The output:

```
Table 51
Chair 53.3
```

Write your answer by following the C++ code structure provided below:

```
class ... {
  private:
    // private member data if any
    . . .
    . . .

  public:
    // public member data if any
    . . .
    . . .

    // directly write public constructor/mutator/accessor
    // implementation here
    type constructor/mutator/accessor {

      . . .
      . . .
    }
};
```

b) Assume that you have correctly defined the `Product` class as requested in question 1 above. Complete the main function below to read the input data of `Product` objects from the "ProductBuy.txt" file and then print out the total price of each product on the screen and to the file named "TotalPrice.txt".

[5 marks]

```
01 int main() {
02   const int LIST_SIZE = 5;
03   string name;
04   double price;
05   int unit, idx = 0;
06
07   Product* pList[LIST_SIZE];
08
09   fstream _______("ProductBuy.txt", ios::in);  // 0.25 marks
10   fstream _______("TotalPrice.txt", ios::out); // 0.25 marks
11
12   if (!fileIn || !fileOut) {
13     cout << "File input/output error!\n";
14     return 1;
15
16   } else {
17     // create an array of Product object (pointer) by using input data
18     // name, price and unit read from the "ProductBuy.txt" file
19     while (fileIn >> _____ >> _____ >> _____) { // 1.5 marks
20       pList[idx] = _____________________________; // 1 mark
21       idx++;
22     }
23
24     // print out product total price to
25     // screen and file ("TotalPrice.txt")
26     cout << "\nPrint out product total price:\n";
27     for (int i = 0; i < idx; i++) {
28       ____________________________________________________;
29       ____________________________________________________;
30       delete (pList[i]);
31     }
32
33     fileIn.close();
34     fileOut.close();
35   }
36 }
```

As a guide below is the content of the "ProductBuy.txt" input file and the output to be produced inside the "TotalPrice.txt" file.

ProductBuy.txt	TotalPrice.txt
Table 25.5 2 Chair 13.30 4 Cup 5.70 4	Table 51 Chair 53.2 Cup 22.8

Question 2 (15 Marks)

Note: *For Question 2, you can write your answer in the question book.*

Consider the following recursive program in C++:

```
01  int sum(int n) {
02    if (n == 0)
03      return 0;
04    else
05      return n + sum(n / 10);
06  }
07
08  int main() {
09    cout << sum(123);
10    return 0;
11  }
```

a) Trace the recursive calls of the sum function in the above program. Write you answer by completing the tracing template below. Fill in the underlined blank spaces/boxes with values ONLY where applicable.

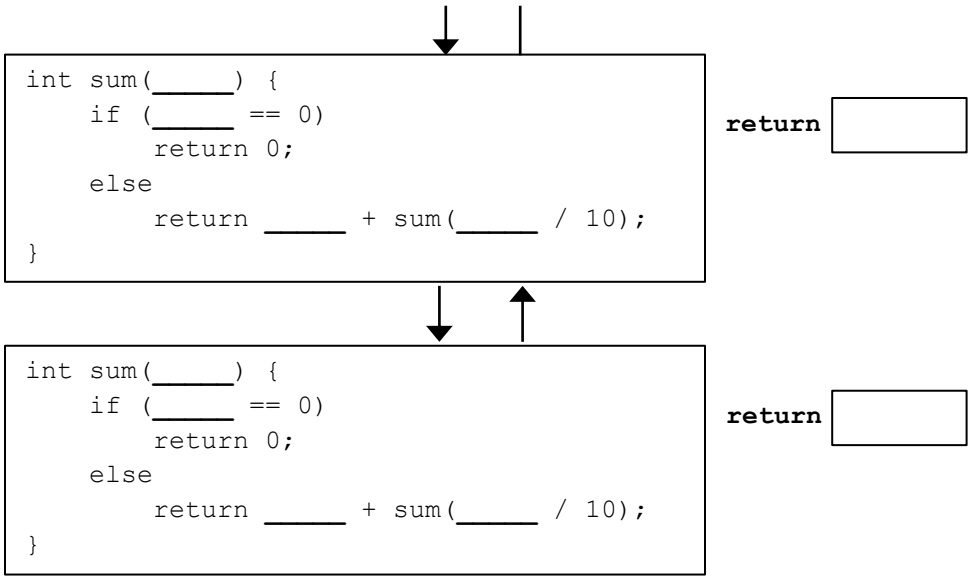
[10 marks]

```
int sum( 123 ) {
    if (_____ == 0)
        return 0;
    else
        return _____ + sum(_____ / 10);
}
```

return ☐

```
int sum(_____) {
    if (_____ == 0)
        return 0;
    else
        return _____ + sum(_____ / 10);
}
```

return ☐



b) What value will be printed by `cout << sum(123)` **:** _____

[2 marks]

c) What is the base/terminal case of the `sum` function: _____

[1.5 marks]

d) What is the general case of the `sum` function: _____

[1.5 marks]

Question 3 (15 Marks)

a) Calculate the number of steps and determine the big-O notation of the following program segment:

[6 marks]

No.	Statement	Number of steps
i.	`int i = 1;`	
ii.	`while (i <= n) {`	
iii.	`i = i * 2;`	
iv.	`}`	
v.	Total steps =	
vi.	Big-O notation =	

b) Calculate the number of steps and determine the big-O notation of the following program segment:

[5 marks]

No.	Statement	Number of steps
i.	`for (int i = 1; i <= n; i++)`	
ii.	`  for (int j = n; j <= i; j--)`	
iii.	`    cout << i * j << "\n";`	
vi.	Total steps =	
v.	Big-O notation =	

c) Arrange the Big-O complexities order of the following functions:

[4 marks]

$\log_3 n$	$\log_2 n$	n^n	n
$n \log_2 n$	n^2	2^n	$2n$

Write your answer in the following format: *lowest* < . . . < . . . < *highest*

Question 4 (20 Marks)

a) A program to sort an array in ascending order using the bubble sort algorithm is given below:

```
01  int temp;
02  int arr[] = {5, 4, 1, 7};
03
04  for (int pass = 1; pass < 4; pass++) {
05    for (int x = 0; x < 3; x++) {
06      if (arr[x] > arr[x+1]) {
07        cout << "swap" << "\n";
08        temp = arr[x];
09        arr[x] = arr[x+1];
10        arr[x+1] = temp;
11      } else {
12        cout << "no swap" << "\n";
13      }
14    }
15  }
```

Complete the table below to trace the values of **`pass`**, **`x`**, **`arr[x] > arr[x+1]`**, the program's output (swap/no swap) and the contents of the **`arr`** array after each pass.

[10 marks]

`pass`	`x`	`arr[x] > arr[x+1]`	**Output**	`arr`
1	0	5 > 4 (true)	swap	{4, 5, 1, 7}
1	1			
1	2			
2	0			
2	1			
2	2			
3	0	1 > 4 (false)	no swap	{1, 4, 5, 7}
3	1	4 > 5 (false)	no swap	{1, 4, 5, 7}
3	2	5 > 7 (false)	no swap	{1, 4, 5, 7}

b) Given the array: {16, 5, 9, 11, 7, 18} to be sorted in an **ascending order**. Complete the **improved version** of **selection sort** analysis table below to determine the followings:

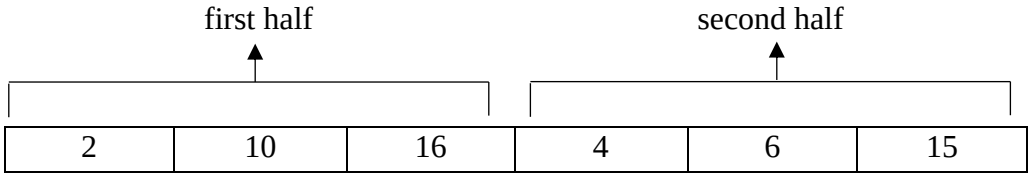
i. Total number of comparison and swap operation in each pass **[4 marks]**

ii. The resulting array after each pass **[6 marks]**

Pass	**Comparison**	**Swap**	**Resulting Array**
1	5	0	{16, 5, 9, 11, 7, 18}
2			
3			
4			
5			

Question 5 (20 Marks)

a) Given two halves of a `list` array variable to be sorted in ascending order:



The array is to be merged and sorted into `tempArray` through `merge(list, 0, 2, 5)` function call. Complete the table below to trace the values of `first1`, `first2`, `list[first1]`, `list[first2]` and the content of `tempArray` until the merge operation is complete.

[10 marks]

first1	**first2**	**list[first1]**	**list[first2]**	**tempArray**
0	3	2	4	{2, -, -, -, -, -}
2	6	16	-	{2, 4, 6, 10, 15, 16}

b) Below is a segment of program to implement the Quick Sort algorithm using the `partition` and `quickSort` functions as follows:

```
01  // Quick Sort function
02  int partition(char array[], int first, int last) {
03    char pivot = array[first];
04    int bottom = first;
05    int top = last;
06    char temp;
07
08    while (true) {
09      while (array[top] > pivot) {
10        top--;
11      }
12      while (array[bottom] < pivot) {
13        bottom++;
14      }
15
16     if (bottom < top) {
```

```
17        temp = array[bottom];
18        array[bottom] = array[top];
19        array[top] = temp;
20
21      } else {
22        return top;
23      }
24  }
25
26
27  void quickSort(char array[], int first, int last) {
28    if (first < last) {
29      int cut = partition(array, first, last);
30      quickSort(array, first, cut);
        quickSort(array, cut + 1, last);
    }
  }
```

Given the array to be sorted by using the above Quick Sort algorithm as follows:

Index	0	1	2	3	4	5
Item	D	E	F	A	C	B

Based on the program and the given array, trace the partitioning process of the Quick Sort algorithm for **`cut = partition(array, 0, 5)`** until the **first cu**t by using item at the first index as pivot. In the trace diagram, show the following steps:

i. The movement of bottom and top
ii. The swapping process: **swap(bottomValue, topValue)**
iii. The cut and the new 2 parts array for next recursive call

[10 marks]

Trace Diagram:

`cut = partition(array, 0, 5);`

pivot value = D **bottom index =** 0 **top index** = 5

Index	0	1	2	3	4	5
Item	D	E	F	A	C	B

swap(D, B)

---

pivot value = D **bottom index =** 1 **top index** = 5

Index	0	1	2	3	4	5
Item	B	E	F	A	C	D

swap(E, D)

---

pivot value = D **bottom index = ___ top index** = ___

Index	0	1	2	3	4	5
Item						

swap(___, ___)

---

pivot value = D **bottom index = ___ top index** = ___

Index	0	1	2	3	4	5
Item						

swap(___, ___)

---

pivot value = D **bottom index = ___ top index** = ___

Index	0	1	2	3	4	5
Item						

swap(___, ___)

---

pivot value = D **bottom index = ___ top index** = ___

Index	0	1	2	3	4	5
Item						

cut point = ___

Next recursive function call:

```
partition(array, ___, ___);
partition(array, ___, ___);
```