UNIVERSITI TEKNOLOGI MALAYSIA

TEST 1
SEMESTER I 2022/2023

CODE OF SUBJECT	**: SECJ2013**
NAME OF SUBJECT	**: Data Structures and Algorithms**
YEAR / COURSE	**: 2SECV, 2SECR, 2SECJ, 2SECB, 2SECP, MJIIT**
TIME	**: 8.00 PM – 10.00 PM (2 Hours)**
DATE	**: Monday, 20 November 2023**
VENUE	**: N28, BK1-BK7, MPK1, MJIIT**

INSTRUCTIONS TO THE STUDENTS:

This question paper consists of 2 parts:

Part A:	20 Objective Questions	20 marks
Part B:	5 Structured Questions	80 marks

ANSWER ALL QUESTIONS IN THE SPACES ALLOCATED IN THIS TEST BOOK.

Name	
Identity card (or matric) Number	
Name of Lecturer	
Subject Code and Section	

This examination book consists of **12** printed pages excluding this page.

PART A – OBJECTIVE QUESTIONS **[20 marks]**

Part A consists of 20 Objective Questions. Choose the correct answer, and write your answer in the answer booklet. Each question carries 1 mark.

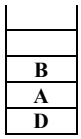
1. Which of the following are the operations applicable for data structures?
   i. To visit each element of an array once for processing
   ii. To remove an existing element from the array
   iii. To insert one or more data elements into an array

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

2. Which of the following is a non-linear data structure?
   i. Arrays
   ii. Linked lists
   iii. Binary tree
   iv. Graph

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

3. Consider the stack as shown in figure below. If 'C' is pushed onto the stack and then the stack is popped, what is at the top of the stack?



   A. A
   B. B
   C. C
   D. D

4. Which of the following is **FALSE** about abstraction in object-oriented programming?

   A. Abstraction is to be implemented on both the data and the functions used to manipulate the data.
   B. Hide both the data and its related functions from programmers.
   C. Combine the data and its related functions into single entity known as class.
   D. Only allow some functions related to data operations to be accessed by the programmers.

5. What is wrong with the implementation of the `Product` class in figure below, in terms of OOP principles?

```
class Product {
  public:
    string name;
    float price;

    void setInfo(string n, float p) {
      name = n;
      price = p;
    }

    void printInfo() {
      cout << name << " - " << price << "\n";
    }
};
```

   A. Not a correct syntax to implement class in C++
   B. No constructor to create the class object
   C. No destructor to delete the class object
   D. Not following the rule of data encapsulation (information hiding)

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

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

   Assume that the class `Person` has the method `getName`. Which of the following is the correct syntax to call the `getName` method of the class by using the declared array?

   A. `pList[3]->getName();`
   B. `pList[0].getName();`
   C. `pList->getName();`
   D. `pList.getName();`

7. Which of the following is **FALSE** about recursion approach?

   A. Implementation of an iteration with a recursion strategy consumes additional memories.
   B. A recursive function can be written in a loop form.
   C. In recursive strategy, a base case is the statement that solves the problem directly.
   D. In recursive approach, the base cases make use of recursion to solve smallest sub problems and combine into a solution to the larger problem.

8. Consider the following mathematical functions.

   `f(x) = 3x  if x<5;` otherwise, `f(x)=f(x-2).`

   If the function is implemented with a recursion strategy, which of the following value of x will fall under the recursion cases?

   A. 2
   B. 4
   C. 3
   D. 5

9. Which of the following code segment implements a recursion strategy?

   A.
```
int f(int m, int n){
    int x=0;
    for (int i=0; i<=m; i++)
    x = x+ n;
}
```

   B.
```
int g(int m, int n){
    return m+n;
}
int f(int m, int n){
    int x=0;
    for (int i=0; i<=n; i++)
    x = g(x,m);
```

   C.
```
int g(int m, int n){
    return m+n;
}
int f(int m, int n){
    int x=0;
    for (int i=m; i<=n; i++)
    g(x,i+i);
}
```

   D.
```
int f(int m, int n){
    if (m==n) return n;
    n+f(m+1,n);
}
```

10. Consider the following code fragment. What is the value of `f(2,4)`?

```
int f(int x, int y)
{
 if (y==0) return x;
 return x + f(x,y-2);
}
```

A. 2
B. 6
C. 8
D. 10

11. The big-O notation for an algorithm with total steps of $n^2 + 4\log_2 n + n^3$ is:

A. n^3
B. n^2
C. $\log_2 n$
D. $4\log_2 n$

12. Which of the following growth-rate functions increases very fast once the problem size increase?

A. $n\log_2 n$
B. n
C. 1
D. $\log_2 n$

13. What is the big-O notation for the following code segment?

```
1  int i , j;
2  i = 3;
3  while ( i<= n ) {
4    for (j = 1 ; j<=n; j++ ){
5      cout <<i<<j;
6  }
7    i = i * 3;
8  }
```

A. $n\log_2 n$
B. $3n\log_3 n$
C. 0
D. $2n + 3n\log_3 n$

14. Which of the following is **TRUE** regarding the complexity of the codes?

```
1  for (i= 2; i <=n;i++){
2     j=3;
3     while (j <=n ){
4          cout<<j<<"\n";
5          j = j*3;
6     }
7  }
```

A. The total steps of the codes is $2n - 2 + 3\log_3 n - 2\log_3 n$.
B. The total steps of the codes can be summarized as $n(n+1)/2$.
C. The complexity of the codes is O($\log_2 n$).
D. The complexity of the codes is O(n^3)

15. Which of the following is **FALSE** about the simple sort technique?

A. Both Selection and Insertion Sort are quadratic algorithms that do not use recursive implementation.
B. The main activities in the sorting process are comparison and swapping. Swapping is related to changing the data position based on the ascending or descending order.
C. Selection sort does not depend on the initial arrangement of data and has the same time complexity for best case and worst case which is O(n^2)
D. Insertion sort works by dividing two parts: the sorted region and the unsorted region where always comparing and swapping adjacent elements in the unsorted region.

16. An array `num` of 6 numbers has the following elements in order:

```
int num[] = {10,9,7,8,6,3}
```

Utilizing the **Selection Sort** to arrange the elements of the array `num` in **ascending order**, what will be the content of the array after the completion of the **second pass** in the sorting process?

A. {7, 9, 10, 8, 6, 3}
B. {3, 9, 7, 8, 6, 10}
C. {7, 8, 6, 3, 9, 10}
D. {3, 6, 7, 8, 9, 10}

17. Based on figure below with sorting implementation, which of the following is **TRUE** for `int array[]={70,60,10,30};`

```
Pass 1: 70 60 30 10
Pass 2: 70 60 30 10

Number of comparisons: 5
Number of swaps: 1
70 60 30 10
```

A. The algorithm that can sort array in descending order is Selection sort
B. The algorithm that can sort array in descending order is Insertion sort
C. The algorithm that can sort array in descending order is Bubble Improve sort
D. The algorithm that can sort array in descending order is Bubble sort

18. Which of the following is/are **TRUE** about merge-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. It always divides the array into equal halves

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

19. How can you determine if an array has been correctly sorted after applying the Quick Sort algorithm?

A. Identify if the pivot is in the middle of the array.
B. Verify that the first and last elements are in the correct order.
C. Check if all elements to the left of the pivot are smaller, and all elements to the right are larger.
D. Determine the number of comparisons performed during the sort.

20. During the partition step of Quick Sort, if the pivot element is 5, and the input array is [3, 8, 4, 1, 5, 7, 9], what will be the position of the pivot element after partitioning?

A. Index 1
B. Index 3
C. Index 4
D. Index 6

PART B - STRUCTURED QUESTIONS **[80 marks]**

Part B consists of 5 structured questions. Answer all questions in the answer booklet. The marks for each part of the question is as indicated.

Question 1

a. Define a `House` class with three (3) member variables (`type`, `price`, `unitSale`) in C++ so it can be used and produced and output as follows:

Example use of the `House` class:

```
House hs("Terrace", 470000, 2);
cout << hs.getType() << " " << hs.getTotalSale() << "\n";
```

The output:

```
Terrace 940000
```

[5 marks]

b. Assume that you have correctly defined the `House` class as requested in **Question 1 a.** above. Complete the main function below at i, ii, iii, iv, v, and vii to read the input data of `House` object from the "UnitSale.txt" file and then print out the total sale of each house type on the screen and to the file named "TotalSale.txt".

```
int main() {
    const int LIST_SIZE = 5;
    string type;
    int price, unitSale, idx = 0;
    House* hSales[LIST_SIZE];

    fstream fileIn((i)_____________, ios::in);
    fstream fileOut((ii)_____________, ios::out);

        if (!fileIn || !fileOut) {
        cout << "File input/output error!\n";
        return 1;

    } else {
       // create an array of House object by using an input data of house
       // type, price and unit sale read from the "UnitSale.txt" file
        while (fileIn >> (iii)_____ >> _____ >> ______) {
            hSales[idx] = (iv)_____________________________;
            idx++;
        }

        cout << "\nPrint out house total sales to screen and file:\n";
        for (int i = 0; i < idx; i++) {
          (v)_____________________________________________________________________;
          (vi)____________________________________________________________________;

          delete(hSales[i]);
        }

        fileIn.close();
        fileOut.close();
    }

    return 0;
}
```

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

UnitSale.txt

```
Terrace 470000 2
Condo 380000 3
Flat 180000 6
```

TotalSale.txt

```
Terrace 940000
Condo 1140000
Flat 1080000
```

[5 marks]

Question 2

Consider the problem of calculating the average of numbers ranging from ***m*** to ***n,*** where ***m*** and ***n*** are of integer values, also, *m* can be greater than *n* and vice versa.

a. Explain how would you use a recursion approach to solve the problem?

[2 marks]

b. Implement the average function in C++ by incorporating a recursion approach.

[9 marks]

c. Trace the execution of calling to the function with values of m and n as follows:
   i. m=3 and n=5
   ii. m=10 and n=7

[4 marks]

Question 3

a. For each of the following expressions, identify the big O notation. Reorder the efficiencies from smallest to largest complexity

No	Expressions	Big O Notation
i	$10 + \log_2 n + 10n$	
ii	$\log n + 15n - 20n + 5n + 10$	
iii	$5 + 2n\log_2 n$	
iv	$7n^2 + 3\log_3 n$	
v	$10+10-20$	

vi Order the efficiencies from smallest to largest complexity:
Answer: _____

[7 marks]

b. Calculate the number of steps and find the big-O notation of the following program segment.

No	Statement	Number of steps
i	`int i, j = 0;`	
ii	`for (i = 1; i <= n; i++) {`	
iii	`  for (j = 1; j < i; j++) {`	
iv	`  cout << " "; }`	
v	`  cout << "";`	
vi	`}`	

vii Total steps =

viii Big O notation =

[8 marks]

Question 4

a. Given the following array named myData, demonstrate the process of sorting the data array into descending order using the selection sort technique.

array index	[0]	[1]	[2]	[3]	[4]	[5]
array item	15	36	3	27	16	12

myData array

Show your answer by using the table format below.

Initial Array		After Pass 1		After Pass 2		After Pass 3		After Pass 4		After Pass 5	
[5]	12	[5]		[5]		[5]		[5]		[5]	
[4]	16	[4]		[4]		[4]		[4]		[4]	
[3]	27	[3]		[3]		[3]		[3]		[3]	
[2]	3	[2]		[2]		[2]		[2]		[2]	
[1]	36	[1]		[1]		[1]		[1]		[1]	
[0]	15	[0]		[0]		[0]		[0]		[0]	

[10 marks]

b. Given an array `int arr[] = {25, 10, 40, 30, 5}` that will be sorted into ascending order. Perform the sorting process using the **Insertion Sort** algorithm. Answer the following:

Original :

25	10	40	30	5

i. Show the result for each pass using the table format below.

Insertion sort

After pass 1					
After pass 2					
After pass 3					
After pass 4					

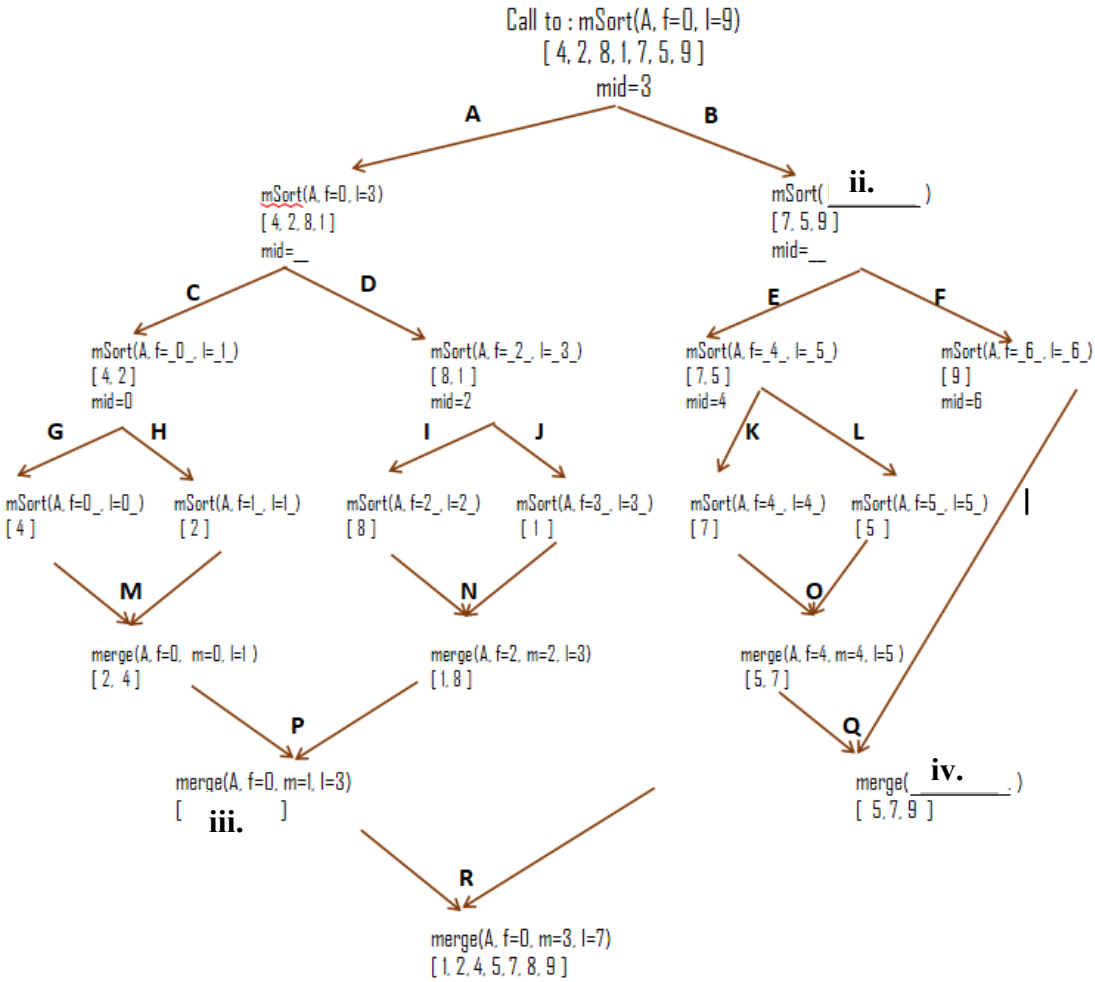
[6 marks]

ii. Give the total number of comparisons and number of insertion for **EACH** pass.

[4 marks]

Question 5

a. The diagram below illustrates how the mergesort algorithm performs sorting on the dataset A=[4, 2, 8, 1, 7, 5, 9].



i. Understand how the mergesort algorithm and the recursive function operates, then write the correct sequence of execution (step A-R) using the format below.

step : A, __, __, __, __, __, __, __, __, __, __, __, __, __, ____, __, __, __ .

[8.5 marks]

ii. Write the correct values of the parameters passed in step B.

[0.5 mark]

iii. Write the value of the data after step P completed.

[0.5 mark]

iv. Write the correct function call in step Q.

[0.5 mark]

b.

array index	[0]	[1]	[2]	[3]	[4]	[5]	[6]
	15	22	46	7	35	18	10

T array

Based on an array called **T**, sort the data in **T** array in ascending order by utilizing the quick sort algorithm. Trace the partitioning process of quick sort for the `cut = partition(T, 0, 6);` until reaching the **first cut point**, using the last item (item at **index 6**) as the **pivot**. In the trace diagram, demonstrate the following steps:

i. The movement of the bottom and top.

[3 marks]

ii. The swapping process during partitioning.

[3 marks]

iii. The cutPoint value and the resulting two-part array for the next recursive call.

[2 marks]

iv. The next two recursive calls for **quicksort** function.

[2 marks]

Trace diagram:

```
cut = partition(T,0,6);
```

pivot value = **initial bottom index =** **initial top index =**

Array index	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**
Array item	15	22	46	7	35	18	10

Current value of **bottom** and **top**

bottom = **top =**

Array index	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**
Array item							

Current value of **bottom** and **top**

bottom = **top =**

Array index	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**
Array item							

Current value of **bottom** and **top**

bottom = **top =**

Array index	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**
Array item							

Current value of **bottom** and **top**

bottom = **top =** **cutPoint =**