PART A – OBJECTIVE QUESTIONS [15 marks]

Part A consists of 10 Objective Questions. Choose the correct answer, and write your answer in page 4. Each question carries 1.5 mark.

1. Khalid and his team are required to develop software for International Student Centre. The system will help the student to manage their registration system that connect to Ministry of Higher Education. Currently, they are conducting preliminary investigation, study the existing system, determine the user requirements and finally provide solution to the problem. In software development process, this phase is called as ________________.

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

2. Which of the following are non- linear structured data types?

   i. Stack
   ii. Graph
   iii. Tree
   iv. Linked list
   v. Queue

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

Answer Question 3 based on the `Student` class declaration in Figure 1. :

```
 class Student {
   public:
      Student (string name = “Mohd Danial”);
      Student (string name, int MC);
   private:
      string name;
      int MC;
      string address
};
```

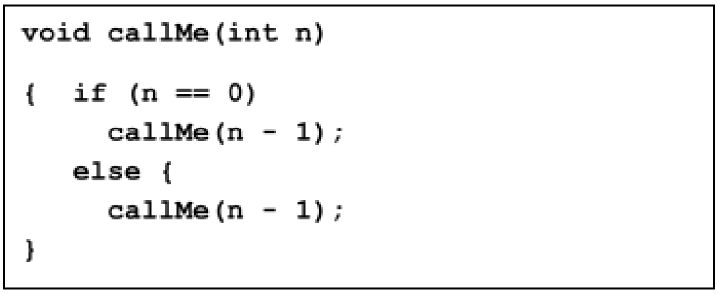
Figure 1. `Student` class declaration.

3. Which of the following object declaration is **INVALID**?

   A. `Student Stud1(5476);`
   B. `Student Stud2(“Khai”);`
   C. `Student Stud3(“Khai”,201701);`

D. **`Student Stud4;`**

4. What is **WRONG** with the recursive function in Figure 2? Assume that we want to stop recursive call when n is equal to 0.



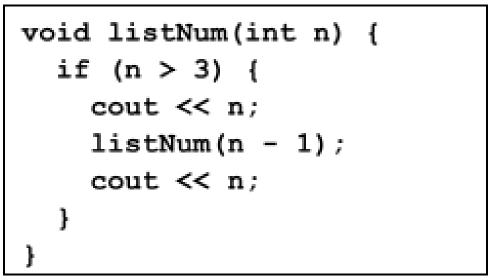
```
void callMe(int n)

{   if (n == 0)
      callMe(n - 1);
    else {
      callMe(n - 1);
}
```

Figure 2. Recursive function.

A. There is no base case defined to stop the recursive calls.
B. There is no general case defined to make the function running in recursive mod.
C. There is base case defined but it continues to make recursive function calls.

D. There is general case defined but not correctly alter the size of the problem.

5. What is the **OUTPUT** of the function in Figure 3, if the statement: **`listNum(5);`** is executed?



```
void listNum(int n) {
  if (n > 3) {
    cout << n;
    listNum(n - 1);
    cout << n;
  }
}
```

Figure 3. Recursive function, **`listNum()`** .

A. 45
B. 54
C. 4554
D. 5445

6. The **big-O notation** for an algorithm with total steps of $\mathbf{9n^2 + 6log_2n}$ is :

   A. $\mathbf{9n^2}$

   B. $\mathbf{n^2 + log_2 n}$

   C. $\mathbf{n^2}$

   D. $\mathbf{log_2n}$

Answer Question 7, based on the codes in Figure 4.

```
for (int a = 1; a <= n; a++
{   for (int b = 1; b<= a; b++)
        cout << b;
    cout << endl;
}
```

Figure 4. Source codes with nested loop

7. Which of the following is **FALSE** regarding the complexity of the codes.

   A. The total steps of the codes is **1 + 2 + 3 + …. + n-1 + n**.

   B. The total steps of the codes can be summarized as **n(n+1)/2**.

   C. The complexity of the codes is $\mathbf{O(log_2 n)}$.

   D. The complexity of the codes is $\mathbf{O(n^2)}$**.**

8. Which of the following growth-rate functions increases very fast and **NOT practical** to be implemented?

   A. **1**

   B. **n**

   C. $\mathbf{2^n}$

   D. $\mathbf{log_2 n}$

9. Which of the following array is able to be divided into a balance or near balance left and right segment in a ***Quick-sort*** partition operation?

   A. Array already sorted and using item at index 0 for pivot value.

   B. Array already sorted and using item at index last for pivot value.

   C. Array can be in any order but the pivot value must be selected using value from the middle index of the array.

   D. Array can be in any order but the pivot value must be selected using the middle value of all the elements in the array.

10. Which of the following operations in **Merge Sort** algorithm performs the sorting process?

A. Dividing.
B. Conquering.
C. Merging.
D. All the above.

Answer sheet for Part A:

Write all your answers for Part A in the space below.

Name: ________________________________

Section: ______________________________

Question No	**Answer**
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	

9.	
10.	
Marks	

PART B - STRUCTURED QUESTIONS **[45 marks]**

Part B consists of 4 structured questions. Answer all questions in the space provided. The marks for each part of the question is as indicated.

Question 1 **[10 marks]**

a) Answer questions (a.i – a.iii) based on **Program 1.1** and **Program 1.2**. The two programs show the code for computing the power of a number where **x** is the base number and **y** is the power ($\mathbf{x}^{y}$), using loop as in **Program 1.1** and recursive call as in **Program 1.2**.

```
// Program 1.1
int calcPower1(int x, int y){
     int result=1;
     while (y!=0){
       result = result*x;
       --y;
     }// end while
     return result;
} // end calcPower1()
//===============================
//main function
int main(int argc, char** argv) {
     cout << calcPower1(2,3);
     return 0;
} // end main()
```

```
// Program 1.2
int calcPower2(int x, int y){
   if (y!=0)
     return
x*calcPower2(x,y-1);
   else
     return 1;
} // end calcPower2()
//
//===============================
===
//main function
int main(int argc, char** argv)
{
     cout << calcPower2(2,3);
```

	`    return 0;`
	`} // end main()`

Program 1.1 and Program 1.2 : Calculate Power

a.i) Write the output for each of the Program.

Output for Program 1.1 : ____________________ (1 mark)

Output for Program 1.2 : ____________________ (1 mark)

a.ii) Show the trace of execution for the recursive call in Program 1.2. (2 marks)

a.iii) Based on the different implementation in Program 1.1 and Program 1.2 for solving the same problem, which algorithm is more efficient in term of memory and time? Give detail explanation. (2 marks) b.i) Trace the flow of the following recursive function, given in **Program 1.3** after it is called through the statement: `cout << helloYou(2, 6);`

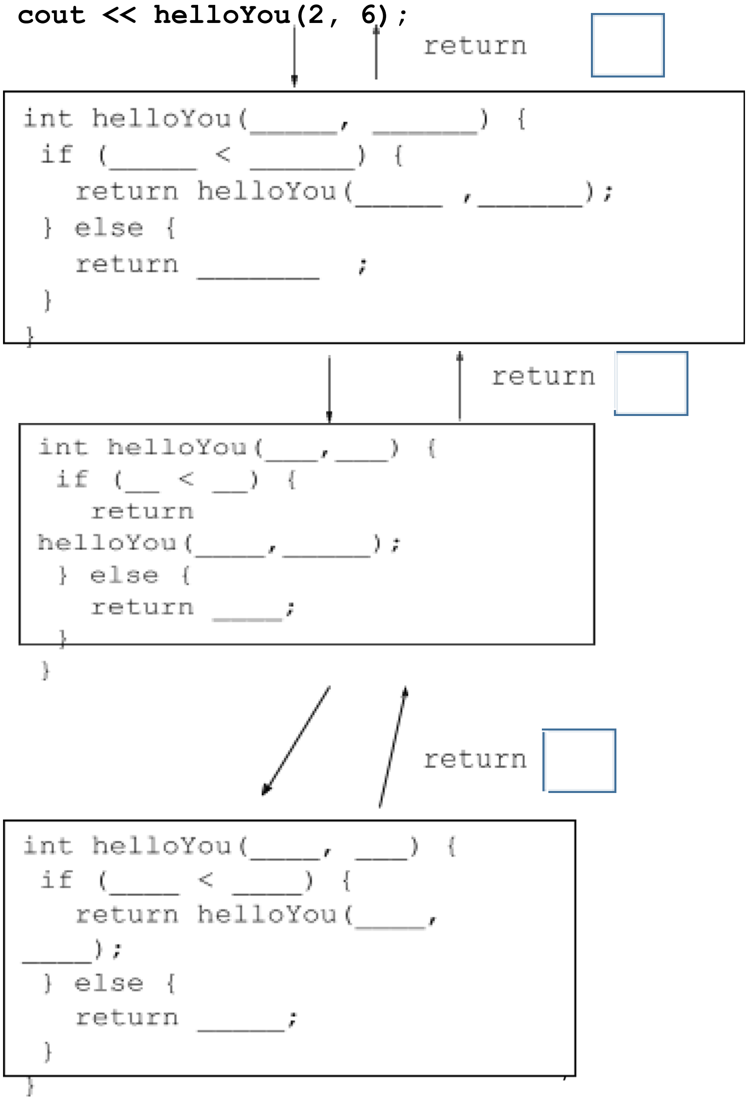
[3 marks]

```
// Program 1.3

int helloYou(int x, int y) {
  if (x < y) {
    return helloYou(x + 1, y - 1);
  } else {
    return x + y;
  }
}
```

Program 1.3 `helloYou`

Write you answer by completing the tracing template below. Fill in the spaces with values **ONLY where applicable**.

`cout << helloYou(2, 6);`



```
return
```

```
int helloYou(_____, _______) {
 if (_____ < _______) {
   return helloYou(_____ ,_______);
 } else {
   return ________  ;
 }
}
```

```
return
```

```
int helloYou(___,___) {
 if (__ < __) {
   return
helloYou(_____,______);
 } else {
   return _____;
  }
}
```

```
return
```

```
int helloYou(____, ___) {
 if (____ < _____) {
   return helloYou(____,
_____);
 } else {
   return ______;
 }
}
```

b.ii) What value will be printed by `cout << helloYou(2, 6);`

Answer : ________________________

(1 mark)

Question 2 **[10 marks]**

a) For each of the following expressions, give the big O notation. Reorder the efficiencies from smallest to largest complexity: [5 marks]

Expressions	**Big O Notation**
$1 + n\log_2(n) + \log_2 n$	
$n + 2n + n^2 + 2n^2$	
$5n + 2\log_2 n$	
$1 + n^3 + 2^n$	
Order the efficiencies from smallest to largest complexity: **Answer:**	

b) Calculate the number of steps and find the big-O notation of the following program segment. [3 marks]

Statements	Number of steps
`for (i= 1; i <=n; i++)`	
`   cout << ″*″;`	
` j = 2;`	
` while (j <=n ){`	
`     cout<< “*****″ << endl;`	
`     j = j*2;`	
` }// end while`	

i. Total steps = ________________ [1mark]

ii. Big O notation = _____________________ [1 mark]

Question 3 **[10 marks]**

Given the following array named, **`data`** , answer the questions in this section.

array index	[0]	[1]	[2]	[3]	[4]	[5]
	8	9	2	4	5	6

`data` array

a) Show the process of sorting the **`data`** array into ascending order using the following sorting techniques:

i. Improved Bubble sort

ii. Selection sort

Show your answer by completing the Table below.

Improved Bubble Sort

		After Pass 1		After Pass 2		After Pass 3	
[5]	6	[5]		[5]		[5]	

[4]	5	[4]		[4]		[4]	
[3]	4	[3]		[3]		[3]	
[2]	2	[2]		[2]		[2]	
[1]	9	[1]		[1]		[1]	
[0]	8	[0]		[0]		[0]	

[3 marks]

Selection Sort

		After Pass 1		After Pass 2		After Pass 3	
[5]	6	[5]		[5]		[5]	
[4]	5	[4]		[4]		[4]	
[3]	4	[3]		[3]		[3]	
[2]	2	[2]		[2]		[2]	
[1]	9	[1]		[1]		[1]	
[0]	8	[0]		[0]		[0]	

After Pass 4		After Pass 5	
[5]		[5]	
[4]		[4]	
[3]		[3]	
[2]		[2]	
[1]		[1]	

[0] □ [0] □ [4 marks]

b) The number of comparison in each pass to sort the data using Improved Bubble Sort and Selection Sort are shown in Table 1. Analyse and discuss the algorithm complexity of both sorting function. Which algorithm has the worst performance and which algorithm has better performance to sort `data` array.

Table 1: Number of Comparison for Bubble Sort

Improved Bubble Sort		**Selection Sort**	
Pass	Number of Comparison	Pass	Number of Comparison
1	**5**	1	**5**
2	**4**	2	**4**
3	**3**	3	**3**
		4	**2**
		5	**1**
Total compares	**12**		**15**

[3 marks]

Question 4 **[15 marks]**

a) Explain the difference between the partitioning/dividing operation in Merge-sort and Quick-sort.

[2 marks]

Given the following **`data2`** array, answer Question b7) and b8).

array index	**`[0]`**	**`[1]`**	**`[2]`**	**`[3]`**	**`[4]`**
	7	8	3	4	5

`data2` array

b) Draw the **Merge-sort** diagram to sort the **`data2`** array into ascending order.

[5 marks]

Program 4.1 is a Quick Sort function which consists of two functions: **`quicksort()`** and **`partition()`**.

```
// Program 4.1
// Quick Sort function
 int partition(int T[], int first,int last)
 {     int pivot, temp;
       int loop, cutPoint, bottom, top;
       pivot= middleValue;
       bottom=first; top= last;
       loop=1;    //always TRUE
       while (loop) {
              while (T[top]>pivot)
              {  top--;          }
              while(T[bottom]<pivot)
              {  bottom++;       }
              if (bottom<top) {
                 temp=T[bottom];
                 T[bottom]=T[top];
                 T[top]=temp;
               }
              else {
                 loop=0;
                 cutPoint = top;
               }
       }//end while
      return cutPoint;
 }
 void quickSort(dataType arrayT[],int first,int
 last)
 {
      int cut;
      if (first<last){
           cut = partition(T, first,last);
           quickSort(T, first,cut);
```

```
                quickSort (T, cut+1, last);
        }
  }

```

Program 4.1 – Quick Sort Algorithm

c) Based on the **`data2`** array, write the ***Quick-sort*** partition work trace for **`cut = partition(data2,0,4)`** until the **first *cut point*** by using item at index 4 (the middle value) as pivot.
In the diagram given, show the following
   i) The movement of bottom and top
   ii) The swapping process
   iii) The cutPoint and the new 2 parts array for next recursive call

array index	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**
	7	8	3	4	5

`data2` array

`cut = partition(data2,0,4)`

`pivot = 5`
`bottom = 0`
`top = 4`
Current value of **`bottom`** and **`top`**

`bottom = ___` **`top = ___`**

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

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

`bottom = ___` **`top = ___`**

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

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

`bottom = ___` **`top = ___`**

Array index	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**

Array item					

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

bottom = ___ **top = ___**

Array index	[0]	[1]	[2]	[3]	[4]
Array item					

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

bottom = ___ **top = ___** **cutPoint = _____**

Array index	[0]	[1]	[2]	[3]	[4]
Array item					

Recursive function and array value after **cutpoint** is returned.

quickSort(data2, __ , __)

quickSort(data2, __, __)

[6 marks]

d) Explain the rationale behind the selection value **5**, which is the item at index 4 of the array as pivot in c)

[2 marks]

================ THE END ==================