LAB 2: RECURSIVE AND ALGORITHM EFFICIENCY

OBJECTIVES

1. Trace the implementation of a recursive function.
   [*Mejejaki perlaksanaan fungi rekursif.*]

2. Write recursive function to solve problems.
   [*Menulis fungsi rekursif untuk menyelesai masalah.*]

3. To analyze the number of steps of algorithms relative to the increasing of input size, n.
   [*Menganalisa bilangan langkah algoritma relatif dengan peningkatan saiz masukan n.*]

4. Find the class of complexity in big 'O' notation.
   [*Mencari kerumitan kelas dalam notasi 'O' besar.*]

LAB EXERCISES

EXERCISE 1: OUTPUT AND PROGRAM TRACING

1. Given the following program, answer the given questions.
   [*Diberi aturcara sebagaimana berikut, jawab soalan yang diberikan.*]

```
 1  // Program 2.1
 2  #include <iostream>
 3  using namespace std;
 4
 5  int fun1(int n)
 6  { if (n<=0)
 7        return 0;
 8   else if (n<=1)
 9        return 1;
10   else
11        return fun1(n-1)+fun1(n-2);
12  }
13
14  int main()
15  {  cout << fun1(5) << endl;
16     system("pause");
17     return 0;
18  }
```

   i. Identify the statement number in function **`fun1()`** according to the following recursive principle.
      [*Kenalpasti nombor bagi penyataan dalam fungsi* ***`fun1()`*** *yang berkait dengan prinsip rekursif berikut.*]
      a. Base Case/Terminal case

[*Kes penamat*]

b. Recursive case
[*Kes rekursif*]

ii. Copy Program 2.1 and run the program. Determine the output of the program.
[*Salin Program 2.1 dan laksanakn aturcara tersebut. Tentukan output aturcara ini.*]

iii. Show the recursive trace for the program.
[*Tunjukkan jejaka rekursif untuk aturcara tersebut.*]

2. Given the following program, answer the given questions.
[*Diberi aturcara sebagaimana berikut, jawab soalan yang diberikan.*]

```
 1  // Program 2.2
 2  #include <iostream>
 3  using namespace std;
 4
 5  int fun2(int, int);
 6  int main ()
 7  {    cout << fun2(7,3);
 8       system("pause");
 9       return 0;
10  }
11
12  int fun2(int x, int n)
13  {   if(n == 0)
14      return 1;
15  else
16  return x * fun2(x, n-1);
17  }
```

i. Copy Program 2.2 and run the program. Write the output for the above program and trace the recursive function.
[*Tulis output yang akan dipaparkan oleh aturcara di atas dan jejaki fungsi rekursifnya.*]

ii. Describe the problem solved by the recursive function and identify the simple solution, recursive process and terminal case.
[*Nyatakan masalah yang diselesaikan oleh fungsi rekursif di atas dan kenalpasti penyelesaian mudah, proses rekursif dan kes penamat.*]

3. Given two recursive functions as followed, answer the given questions.
[*Diberi dua fungsi rekursif berikut, jawab soalan yang diberikan.*]

```
 1  // Program 2.3
 2  void function_01(int n)
 3  {
 4      cout << "Calling function_01\n";
 5      if (n < 5)
 6          function_01(n - 1);
 7  }
```

```
1  // Program 2.4
2  void function_02(int n)
3  {
4      if (n > 1)
5          function_02(n - 2);
6      cout << n << "  ";
7  }
```

i. Write two complete programs, one program will call `function_01()`with parameter 3 and the other one will call `function_02()`with parameter 3.
[*Tulis dua aturcara lengkap, satu aturcara akan memanggil fungsi* `function_01()`*dengan parameter 3 dan yang satu lagi akan memanggil fungsi* `function_02()` *dengan parameter 3.*]

ii. Identify problem in one of the programs.
[*Kenal pasti masalah di dalam salah satu aturcara-aturcara tersebut.*]

iii. Show the recursive trace of the error free program.
[*Tunjukkan jejakan recursif untuk aturcara yang tiada ralat.*]

4. Given the following programs, copy and run the programs. Identify the output of the programs and trace the recursive functions.
[*Diberi aturcara berikut, salin dan laksanakan aturcara-aturcara tersebut. Kenalpasti output yang akan dipaparkan oleh aturcara-aturcara di bawah dan jejaki fungsi rekursifnya*]

i. Function to print integers.
[*Fungsi yang mencetak integer.*]

```
1  // Program 2.5
2  #include <iostream>
3  using namespace std;
4
5  void printIntegers(int n);
6  int main()
7  { int number;
8        cout<<"\nEnter an integer value :";
9        cin >> number;
10       printIntegers(number);
11       system("pause");
12      return 0;
13 }
14 void printIntegers (int nom)
15 {    if (nom >= 1)
16           cout << "\Value : " << nom<<endl;
17      printIntegers (nom-2);
18 }
```

ii. Greatest Common Divisor (GCD) Function
[*Fungsi Pembahagi Sepunya Terbesar.*]

```
 1  // Program 2.6
 2  #include <iostream>
 3  using namespace std;
 4
 5  int GCD(int a, int b);
 6  int main()
 7  {   int first=3, second=8;
 8      cout<<GCD(first,second)<<endl;
 9        system("pause");
10       return 0;
11  }
12
13  int GCD(int a, int b)
14  {
15     if (a % b == 0) // BASE CASE
16        return b;
17     else  // RECURSIVE
18        return GCD(b, a%b);
19  }
```

iii. Product function
[*Fungsi darab.*]

```
 1  // Program 2.7
 2  #include <iostream>
 3  using namespace std;
 4
 5  int Calc (int n)
 6  {
 7     if (n < 0)
 8       return n;
 9     else
10       return Calc(n-1)* Calc(n-2);
11  }
12  int main()
13  {  cout << Calc(5) << endl;
14     system("pause");
15     return 0;
16  }
```

EXERCISE 2: PROBLEM SOLVING USING LOOP AND RECURSIVE FUNCTION

1. Given the following program, answer the given questions.
[*Diberi aturcara sebagaimana berikut, jawab soalan yang diberikan.*]

```
 1  // Program 2.8
 2  int function_03(int n)
 3  {
 4       int total=0;
 5       while (n > 0)
 6       {  cout << n << " ";
 7          total += n;
 8          n--;
```

```
 9       }
10       return total;
11  }
```

i. Write **`main()`** function which calls **`function_03()`** with parameter equal to 5.
[*Tuliskan fungsi* **`main()`** *yang akan memanggil fungsi* **`function_03()`** *dengan parameter 5* .]

ii. Rewrite the implementation of **`function_03()`** using recursive approach.
[*Tulis semula implementasi bagi* **`function_03()`** *secara rekursif.*]

EXERCISE 3: NUMBER OF STEPS

1. Calculate the number of executions for every statement in the program below. Variable **n** in the program represents the number of input. Find the total executions and determine the time complexity (O **notation)** for every program in this section.
[*Kira bilangan perlaksanaan bagi setiap pernyataan dalam setiap keratan aturcara di bawah. Pembolehubah* ***n*** *mewakili bilangan input. Seterusnya dapatkan jumlah perlaksanaan dan tentukan masa kerumitan (dalam* ***notasi O****) untuk setiap keratan aturcara pada bahagian ini.]*

i.

```
1  // Program 2.9
2  for (i=2; i<n; i++)
3      cout << i;
4  for (j=1; j<=n; j=j*2)
5      cout << j;
6
```

ii.

```
1  // Program 2.10
2  int i=1;
3  while (i<=n)
4  {  for (j=1; j<=i; j++)
5       cout << i << j;
6     i++;
7  }
```

iii.

```
1  // Program 2.11
2  int i, j=1;
3  for(i=1; i<=n; i++){
4        j = j*2;
5   }
6  for (i=1; i<= j; i++) {
7          cout << j << "\n";
8  }
```

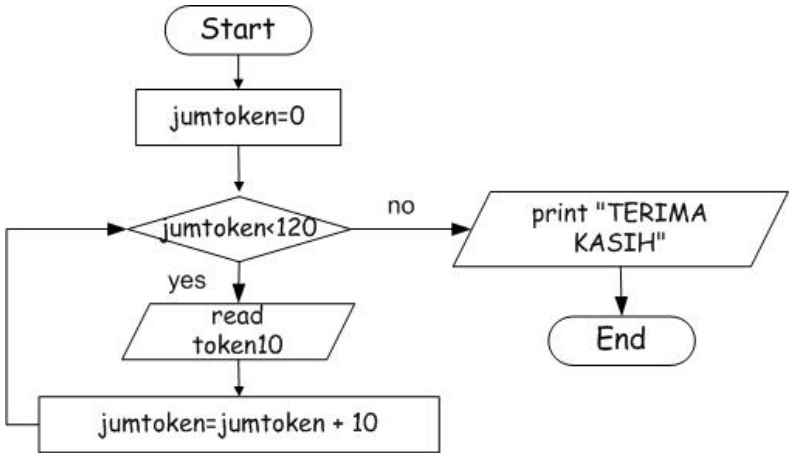
iv.

```
1  // Program 2.12
2  void a(int n)
3  {  for (int a=3; a<=n; a++)
4       for (int a=1; a<n; a++)
5          cout<<a;
6  }
```

v.

```
1  // Program 2.13
2  for (int a=3; a<=n; a++)
3       for (int a=1; a<n; a++)
4          cout<<a;
```

2. Given the following flow chart.
[*Diberikan carta alir berikut.*]



vi. Calculate the **number of steps** and find the big-O notation of the above algorithm.
[*Kira **bilangan langkah** dan dapatkan notasi O besar bagi algoritma di atas.*]

EXERCISE 4: ALGORITHM ANALYSIS

1. Copy the following program. Complete, compile and run the program [*Salin aturcara aturcra berikut. Lengkapkan, kompil dan laksanakan aturcara tersebut.*]

```
 1  // program 2.14
 2  #include <iostream.h>
 3
 4  void fungsi_1();
 5  void fungsi_2(int);
 6  void fungsi_3(int);
 7  void fungsi_4(int);
 8  void fungsi_5(int);
 9  void fungsi_6(int);
10  void fungsi_7(int);
11  void fungsi_8(int);
12
13  int main() {
14        int choice;
15        int input;
16
17        cout << "Analysis the number of steps of
18  algorithms \n\n";
19
20        cout << "choose fungsi [1 - 8]: ";
21        cin >> choice;
22
23        if (choice == 1) {
24              fungsi_1();
25              cout << "\n";
26
27        } else {
28              cout << "Masukkan bil. input: ";
29              cin >> input;
30
31              switch (choice) {
32
33                    case 2:
34                          fungsi_2(input);
35                          break;
36
37                    //please complete this line statement
38
39                    case 8:
40                          fungsi_8(input);
41                          break;
42              }
43        }
44
45        return 0;
46  }
47
48  void fungsi_1() {
49        int counter = 1;
50        cout << "Arahan cout kali ke " << counter << "\n";
51  }
52
53
54  void fungsi_2(int n) {
```

```
 55       int counter = 1;
 56       int i = 0;
 57
 58       for (i = 1; i <= n; i++) {
 59         cout << "Arahan cout kali ke " << counter <<
 60 "\n";
 61             counter++;
 62       }
 63 }
 64
 65 void fungsi_3(int n) {
 66       int counter = 1;
 67       int i = 0;
 68       int j = 0;
 69
 70       for (i = 1; i <= n; i++) {
 71             for (j = 1; j <= n; j++) {
 72                cout <<"Arahan cout kali ke " <<counter<<
 73 "\n";
 74                 counter++;
 75             }
 76       }
 77 }
 78
 79 void fungsi_4(int n) {
 80       int counter = 1;
 81       int i = 0;
 82       int j = 0;
 83       int k = 0;
 84
 85       for (i = 1; i <= n; i++) {
 86             for (j = 1; j <= n; j++) {
 87                   for (k = 1; k <= n; k++) {
 88       cout << "Arahan cout kali ke " << counter << "\n";
 89                         counter++;
 90                   }
 91             }
 92       }
 93 }
 94
 95
 96 void fungsi_5(int n) {
 97       int counter = 1;
 98       int i = 0;
 99
100       for (i = 2; i <= n; i = i * 2) {
101          cout << "Arahan cout kali ke " << counter <<
102 "\n";
103             counter++;
104       }
105 }
106
107 void fungsi_6(int n) {
108       int counter = 1;
109       int i = 0;
120
121       for (i = 2; i <= n; i = i * 4) {
122        cout << "Arahan cout kali ke " << counter <<
123 "\n";
124             counter++;
125       }
```

```
126 }
127
128 void fungsi_7(int n) {
129       int counter = 1;
130       int i = 0;
121       int j = 0;
122
123       for (i = 2; i <= n; i = i * 2) {
124             j = 1;
125
126             while (j <= n) {
127              cout << "Arahan cout kali ke "<<counter <<
128 "\n";
129                   counter++;
130                   j++;
131             }
132       }
133 }
134
135 void fungsi_8(int n) {
136       int counter = 1;
137       int i = 1;
138       int j = 1;
139
140       while (i <= n) {
141             j = j * 2;
142             i++;
143       }
144
145       for (i = 1; i <= j; i++) {
146        cout <<"Arahan cout kali ke " << counter << "\n";
147             counter++;
148       }
149 }
```

i. For `fungsi_2` to `fungsi_8`, try varying the value *n* (refer to the following table).
[*Bagi `fungsi_2` hingga `fungsi_8`, cuba naiali n yang pelbagai (rujuk jadual berikut).*]

Function	**n**
`fungsi_2`	1, 2, 3, 4, 5
`fungsi_3`	1, 2, 3
`fungsi_4`	1, 2, 3
`fungsi_5`	4, 8, 16, 32
`fungsi_6`	4, 8, 16, 64, 256
`fungsi_7`	4, 8, 16
`fungsi_8`	1, 2, 3, 4, 5

ii. Write the result in the following table.
[*Tulis keputusan dalam jadual berikut.*]

Function name	**n**	**Number of output**
`fungsi_2`	1	1
`fungsi_2`	2	2

. . .	...	...
. . .	...	...
. . .	...	...
fungsi_8	5	???

EXERCISE 4: PROBLEM SOLVING

1. Given the following pattern.
[*Diberi cetakan dibawah.*]

```
# # # # # #
 # # # # #
  # # # #
   # # #
    # #
     #
    # #
   # # #
  # # # #
 # # # # #
# # # # # #
```

iii. Write a complete C++ program using recursive function that prompts the user to enter a non-negative integer and generate the above pattern. The above pattern is generated with input 6.
[*Tulis satu aturcara C++ lengkap yang menggunakan fungsi rekursif, meminta pengguna memasukkan integer bukan negatif dan mencetak cetakan output di atas. Cetakan di atas dihasilkan dengan input 6.*]

iv. Change the recursive function with control loop.
[*Tukar fungsi rekusif dengan menggunakan gelung kawalan.*]

v. Find the total executions and determine the time complexity (O **notation)** for solution Question ii.
[*Dapatkan jumlah perlaksanaan dan tentukan masa kerumitan (dalam **notasi O**) untuk penyelesaian Soalan ii.*]

2. Write a complete C++ program using recursive function that takes n as an integer parameter and P(n) is the recursive function with the following definition.
[*Tuliskan satu aturcara yang mempunyai fungsi rekursif dengan menganggapkan n adalah suatu parameter integer dan P(n) adalah satu fungsi rekursif dengan takrifan:*]

$$P(n) = \begin{cases} 3 * n & \text{if } n < 5 \\ P(n-2) * 4 + P(n-1) * 2 & \text{otherwise} \end{cases}$$

i. Run the program with the following function's parameters.
[*Laksanakan aturcara tersebut dengan parameter-parameter berikut.*]

a. P(2)
b. P(8)
c. P(10)

3. Write a complete C++ program to count the number of palindrome word in a text file. Palindrome word is a word that reads the same forward and backward. Examples of the word in Malay are "kayak" and "masam". The program uses recursive function to determine a string argument is a palindrome word and the function returns `bool`.
[*Tuliskan satu aturcara C++untuk mengira bilangan perkataan palindrom dalam satu fail teks. Perkataan palindrom adalah perkataan yang dibaca sama kedepan dan ke belakang. Contoh perkataan palindrom dalam bahasa melayu adalah "kayak" dan "masam". Aturcara ini menggunkan satu fungsi rekursif untuk menentukan argumen rentetan adalah perkataan palindrom dan ia akn memulangkan* `bool`.]