1. Precisely define "Algorithm". Give one consequence of the definition. *[3] An algorithm is an ordered set of unambiguous executable steps which defines a terminating process*

2. What "grouping" do the following algorithms belong to? *[3]*

   (a) quicksort *divide and conquer*

   (b) shortest path in a graph (with positive weights) *greedy*

   (c) multiplication of n matrices of uneven sizes *dynamic programming*

3. How is the problem of ambiguity of algorithm representation usually solved? *[2]*
   By defining a precise set of primitives (building blocks) and composition rules

4. Give an example (by name) of an algorithm in each of the following classes: $\Theta(\lg n)$, $\Theta(n)$, $\Theta(n^2)$. *[3]*
   ex: Binary search, linear search, selection sort.

5. This question concerns the following pseudo-code:

```
z← 0;
x← 1;
while (x < 6) do
   (z ← z+x;
    x ← x+1)
```

   (a) convert the pseudo-code to a C code fragment which uses the same constructs. No need to declare your variables. *[2]*

```
z = 0;
x = 1;
while (x < 6) {
   z = z+x;
   x = x+1;
}
```

   (b) convert the pseudo-code to a C code fragment that uses a *for* loop. No need to declare your variables. *[2]*

for (z=0,x=1; x < 6; z+=x, x++) ;

or

z = 0; for (x=1; x < 6; x++)   z = z + x;

6. Design an algorithm that, given two strings of characters, tests whether the first string appears somewhere in the second. You may assume that neither strings are empty. Write the algorithm in C. You may use the C library function *strlen* in your implementation. *[5]*

```
/* is s1 contained in s2 ? */
bool contains(char *s1, char *s2) {
    int l1, l2,i,j;
char *s;
bool same;

l1 = strlen(s1);
l2 = strlen(s2);
if (l1>l2) {
    return false;
} else {
    /* for each possible starting point in s2 */
    for (i=0,s=s2; i<=l2-l1; i++, s++) {
    same = true;
j = 0;
/* if s1 and s are the same */
while (j<l1 && same) {
    if (s[j] == s1[j])
    j++;
else
    same = false;
}
if (j==l1 && same) /* found a match! */
    return true;
}
return false;
}
}
```

Note that the code above uses the C99 standard version of C.

7. What is the difference between an assembler and a compiler? *[2]*
An assembler translates from assembly language to machine language whereas a compiler translates from a machine-independent high-level language to machine language.

8. What is the difference between a declarative statement and an imperative statement? *[2]*
   Declarative statements simply declare the type of variables whereas imperative statements describe actions

9. Give 3 different, syntactically correct examples of control structures in C. *[3]*
   for loop, while loop, do loop, assignment, if statement, switch statement.

10. What is the difference between an object and a class? *[2]*
    An object is an instance of a class. In other words, a class defines a "type" of object.

11. What is a constructor? *[1]*
    A method (or function) in a class with the same name as the class which constructs (builds) an instance of an object in that class.

12. Name the 4 programming paradigms. Pick 2 and describe them. *[4]*

    - imperative (procedural) - algorithm-centric
    - OO - data-centric
    - functional - function-centric
    - logic (declarative) - problem-centric

13. Define *encapsulation* in the context of imperative programming languages. Define *encapsulation* in the context of object-oriented programming languages. Does the assigned meaning match the structuring methodology of each paradigm? Explain. *[6] In the*

    context of imperative programming, encapsulation is to hide an algorithm by putting in inside a function (or procedure) [instead of having the code in-line in another function/procedure]. In the context of object-oriented programming, encapsulation is to hide the details of the implementation of some data (and its manipulation) by grouping it in a class [instead of having the data-structure and its manipulation known by every piece of code that needs it]. This meaning exactly matches the structuring methodology of each paradigm (algorithm-centric and data-centric respectively).

14. What is the run-time of these code fragments, as a function of n? Assume the proper declarations have been made and that the code is correct. Use $\Theta()$ notation for your answers: *[6]*

```
for $(i = 0; i < n; i++)$ {
   for $(j = 0; j < n; j++)$ {
      $a[i][j] = i + j;$
   }
}
```

$\Omega(n^2)$

```
for $(i = 0; i < n; i++)$ {
   for $(j = 0; j \leq 10; j++)$ {
      $a[i][j] = i - j;$
   }
}
```

$\Omega(n)$

```
for $(i = 0; i < n; i++)$ {
   for $(j = 0; j < n; j++)$ {
      $a[i][j] = i + j;$
   }
   for $(j = 0; j < n; j++)$ {
      $a[i][j] += a[j][i];$
   }
}
```

$\Omega(n^2)$

15. Given the following XML:

```
<courses>
    <course>
        <name>CS 1MD3</name>
    </course>
    <course>
        <attendance>84</attendance>
        <course>
            <name>SE 3M04</name>
        </course>
    </course>
</courses>
```

   (a) Is it well-formed ? *[1]. yes*
   (b) Is it valid ? *[1]. it is not posible to tell*

16. Where would you find ASTs and parse trees? *[1]*
compiler

17. SVG and postscript use the same technique to represent, respectively, 2D graphics and printer instructions. What technique do they use? *[2]*
in both, the data is represented using programs

18. Rewrite the following C fragment to use a *case* statement. *[4]*

```
extern int f1(int);
extern int f2(int);
extern int f3(int);
extern int f4(int);

typedef int (*pifi)(int);

static pifi tab[7] = {f2, f3, f1, f1, f1, f4, f2};

int g(int i) {
    return tab[(i*i)%7](i);
}

int g(int i) {
    int res;

    switch ((i*i)%7) {
    case 0: res = f2(i); break;
    case 1: res = f3(i); break;
    case 2: res = f1(i); break;
    case 3: res = f1(i); break;
    case 4: res = f1(i); break;
    case 5: res = f4(i); break;
    case 6: res = f2(i); break;
    }
    return res;
}
```

END TEST