UNIVERSITY OF KWAZULU-NATAL
Discipline of Electrical, Electronic and Computer Engineering
(Howard College Campus)

Examinations: **November 2015**

ENEL3SF H2 : Software Engineering II

DURATION :2 HOURS MARKS : 100

Examiners:

Prof. J.R. Tapamo

Dr. T. Quazi

Instructions:

1. Answer ALL 5 questions.
2. Start each question on a new page, and number all your answers clearly.
3. Calculators may be used.
4. Ensure you have 6 pages, including this one.

Question 1: *Software Project Management and Planning* **[16 marks]**

(a) Software project management includes [2]

(A) staffing

(B) training

(C) estimation

(D) all of the above

(E) none of the above

(b) If team size is doubled at the beginning of the project, [2]

(A) the development time may be reduced

(B) the development time depends of the nature of the project

(C) the development time may be increased

(D) the development time is unaffected

(E) none of the above

(c) The most basic static variable for estimating software development efforts is [2]

(A) software size

(B) team size

(C) user community size

(D) all of the above

(E) none of the above

(d) Which of the following need not be looked into by software project managers? [2]

(A) Legal liabilities of software development

(B) Effort estimation of the project

(C) Effort distribution of the project

(D) all of the above

(E) none of the above

(e) Explain why the process of project planning is iterative and why a plan must be continually reviewed during a software project. [8]

Question 2: *Software Metrics and Quality Assurance* **[22 marks]**

(a) Software measurement helps in [2]

- (A) software maintenance
- (B) software planning
- (C) software delivery
- (D) software packaging
- (E) all of the above

(b) LOC computes [2]

- (A) number of lines in the source code of the software
- (B) number of logical lines in source code
- (C) number of commented and uncommented lines in source code
- (D) all of the above
- (E) none of the above

(c) Minimum cyclomatic complexity of any compilable code can be [2]

- (A) -1
- (B) 0
- (C) 1
- (D) 0.1
- (E) Never predicted without a given program code

(d) Token is [2]

- (A) an operator
- (B) an operand
- (C) a keyword
- (D) a literal
- (E) all of the above

(e) The source code in Figure 1 is a C implementation a searching method that searches **key** in an array **a** and returns its position if it is found and -1 otherwise.

```
int f(int a[], int key)
{
  int i = 0;
  while (i < SIZE)
  {
    if(a[i] == key) return i;
    i = i + 1;
  }
  return -1;
}
```

Figure 1: Searching method

(e.1) List all distinct operators and all distinct operands. [8]

(e.2) Give the following metrics of the program: the number of distinct operators, the number of distinct operands, the program length and the program vocabulary size. [6]

Question 3: *Project Scheduling* **[16 marks]**

Consider the following project schedule

Activity ID	Predecessor	Duration *(weeks)*
A	NONE	4
B	A	3
C	A	10
D	B	4
E	D,C	3
F	E	2
G	F	5
H	E	6

(a) Draw a Gantt chart for the above project schedule. [2]

(b) Draw a PERT diagram for the above project schedule. [2]

(c) What is the earliest completion time for the project? What is the Slack Time of each task? What is the critical path of the project? [10]

(d) What would happen to the project if task F was delayed by 4 weeks? [2]

Question 4: *Software Implementation and Testing* **[38 marks]**

(a) What white-box testing? In which situation is it normally used? [5]

(b) Explain why testing can only detect the presence of errors, not their absence. [4]

(c) The C function, in Figure 2, sorts an array **arr** of size **n**. The array and its size are passed as arguments.

```
void Sorting(int arr[], int n)
{
 int i,j;
 for(i = 0;i < n-1;i++)
 { int imin = i;
   for(j=i+1;j < n;j++)
    if (arr[imin] > arr[j])
    imin =j;
    int temp       = arr[imin];
        arr[imin] = arr[i];
        arr[i]    = temp;
 }
}
```

Figure 2: Sorting in increasing order

(c.1) Draw a control-flow graph of this code. What will be its cyclomatic complexity, $V(G)$? [4]

(c.2) Propose test cases that achieve statement coverage. [2]

(c.3) Propose test cases that achieve edge coverage. [2]

(d) Consider the problem of computing the value of a polynomial of the third order:

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

What is the number of test cases one should develop for exhaustive testing assuming that one uses 32 bits to represent real number coefficients? Assuming that each test case uses up to $8 \mu sec$. What time is needed to perform this testing. [6].

(e) Write a function **Ranking** in C/C++ that receives as input two arrays of integers, a and p, and their size N and modifies the values in p based on a, in such a way that for all $i = 0, 2, \ldots, N-1$, the value in $p[i]$ is the rank of $a[i]$, when the ranking in increasing order of elements of a is considered. In other words **Ranking** inspects a, and records the ranking (rank in increasing order) of elements of a in p. For instance, If the input of sorting is $a = \{7, -2, -7, 4\}$, $N = 4$ and $p = \{0, 0, 0, 0\}$, after the execution of **Ranking**, we will have $p = \{2, 1, 3, 0\}$. Develop a set of test cases that you feel will adequately test this program. [15]

Question 5: *Embedded Systems* **[8 marks]**

(a) What is the key difference between the Observe and React pattern and the Environmental Control pattern? [3]

(b) What is a circular buffer and how is it used in real-time systems? [5]