AUTUMN MID SEMESTER EXAMINATION-2023

School of Computer Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Software Engineering
[IT-3003]

Time: 1 1/2 Hours **Full Mark: 20**

Answer Any four Questions including Question No. 1 which is compulsory.
The figures in the margin indicate full marks. Candidates are required to give theiranswers in their own words as far as practicable and all parts of a question should beanswered at one place only.

1. Answer all the questions. [2 x 5]

a) What is the effect of pair programming on the development of user stories in the XP model?

"Pair programming" is not so much a specific technique to pair, but more of a mindset to have about pairing. The development of a user story or a feature usually requires not just coding, but many other tasks. As a pair, you're responsible for all of those things.

A developer typically writes code alone. Pair programming is a practice in which two developers are paired together to jointly complete a task. The task could be software design, algorithm, coding or testing. The rationale is that two minds are better than one. If done correctly, pair programming yields better software faster and at lower cost.

b) Do you agree with the following statement (Mention True/False)—The focus of exploratory programming is error correction while the software engineering principles emphasis error prevention"? Give the justifications behind your answer.

An important difference is that the exploratory software development style is based on error correction while the software engineering principles are primarily based on error prevention. In the exploratory style, coding was considered synonymous with software development. Developing a software product with exploratory programming style
believed in developing a working system as quickly as possible and then successively modifying it until it performed satisfactorily. In the modern software engineering,
coding is regarded as only a small part of the overall software development activities. There are several development activities such as design and testing which typically require much more effort than coding.

c) Discuss the Agile Manifesto/Philosophy.

The four core values of Agile software development as stated in the Agile Manifesto are as follows:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a project plan.

The following are the 12 principles of the Agile Manifesto:

- ✓ Meeting end users' needs with early and continuous delivery of work.
- ✓ Being open to changes in requirements even late in the project.
- ✓ Delivering completed work at regular intervals, preferably short ones.
- ✓ Working with the project team and business owners daily.
- ✓ Assembling a motivated team, providing them with the right environment and support, and trusting them.
- ✓ Communicating face-to-face regularly.
- ✓ Using completed work to measure progress.
- ✓ Creating processes that promote sustainable efforts and a constant pace of work.
- ✓ Requiring continuous attention to excellence through good design.
- ✓ Encouraging simplicity.
- ✓ Recognizing that the best work emerges from self-organized teams that deliver the best architectures and designs.
- ✓ Reflecting regularly on how the team can be more effective and fine-tuning and adjusting the approach.

d) What is the difference between the functional and the non-functional requirements of a system? Identify at least two functional requirements of online Railway Reservation System.

Functional Requirements	Non Functional Requirements
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies "What should the software system do?"	It places constraints on "How should the software system fulfill the functional requirements?"
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
Usually easy to define.	Usually more difficult to define.
Example **1)** Authentication of user whenever he/she logs into the system. **2)** System shutdown in case of a cyber attack. **3)** A Verification email is sent to	**Example** **1)** Emails should be sent with a latency of no greater than 12 hours from such an activity. **2)** The processing of each request should be done within 10 seconds

e) What do you mean by project size? What are the popular metrics to measure project size?
Accurate estimation of the problem size is fundamental to satisfactory estimation of effort, time duration and cost of a software project. In order to be able to accurately estimate the project size, some important metrics should be defined in terms of which the project size can be expressed. The size of a problem is obviously not the number of bytes that the source code occupies. It is neither the byte size of the executable code. The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.

- Length (LOC)
- Functionality
- Complexity

2. (a). Briefly explain the important differences and similarities between the incremental and evolutionary models of SDLCs.

In the Evolutionary model, the complete cycle of activities is repeated for each version, whereas in the Incremental model, the User Requirements Definition, System Requirements Definition, and System Design/Architecture activities are factored out of the sequence of incremental deliveries and occur only once, at the outset of the project. This distinction is important. It means that the sum of all increments represents the totality of a single system. It means that the sum of all increments represents the totality of a single system, which must be analyzed and designed once at the start of the project. Thereafter, the physical increments are individually designed, tested, and delivered at successive points in time.

The aspect of requirements analysis and design once at the start of the project is not present in the evolutionary model in which the coupling between successive versions is much looser. Indeed, in the evolutionary model, compatibility between successive versions, although desirable, is by no means assured. In the incremental model, on the other hand, compatibility between successive increments is de rigueur. [5 Marks]

(b). What do you understand by the problems of overspecification, inconsistent requirements, and noise in an SRS document? Explain each of these with suitable examples. [5 Marks]

Over specification

It occurs when the analyst tries to address the "how to" aspects in the SRS document. It limits the imagination of the developers/designers to come up with a a good solution. For example, in a movie database application, you don't need to specify how the movies are stored in the database, and which algorithm you're using to fetch them.

Noise

Noise refers to the presence of material not directly relevant to the software development process. It is hardly of any use to the software developers and would unnecessarily clutter the SRS document, diverting the attention from the crucial points. For example, In the register customer

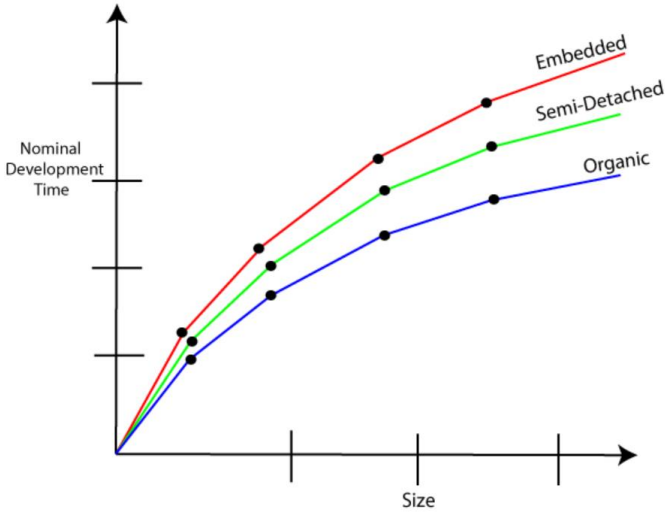
function, information like, who mans the customer registration department and at what time do they work, etc., are considered as noise.

Inconsistent

Parts of the requirements negate information in other requirements. Example: One customer says turn off heater and open water shower when temperature > 100 C. Another customer says turn off heater and turn ON cooler when temperature > 100 C

3. (a). "According to the COCOMO model, if the size of a software is increased by two times, the time to develop the product usually increases by less than two times", justfy your answer.

When the size of the product increases two times, the time to develop does not double but rises moderately.



Development time versus size

The development time versus the product size in KLOC is plotted in fig. From fig it can be observed that the development time is a sub linear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately. This can be explained by the fact that for larger products, a larger number of activities which can be carried out concurrently can be identified. The parallel activities can be carried out simultaneously by the engineers. This reduces the time to complete the project. Further, from fig, it can be observed that the development time is roughly the same for all three categories of products. For example, a 60 KLOC program can be developed in approximately 18 months, regardless of whether it is of organic, semidetached, or embedded type.

[5 Marks]

(b). Calculate Function point for an upcoming online healthcare website project which has the following function and feature details -

Number of User Inputs: 40, Number of User outputs: 20, Number of user enquiries: 20. The data will be stored and retrieved from the patient file, medicine details file and order derails (transaction) file. The application has two external interfaces for the payment.

From the above details, 20 user inputs are simple and remaining inputs are complex. All outputs are complex. 10 user enquiries are simple searches and hence, can be considered to be as simple. Remaining enquiries are complex. All data-structures/logical files are complex. Both the interfaces are simple. All 14 influencing parameters have moderate impact. Hence, Degree of influence is 42. [5 Marks]

To calculate the Function Point (FP) for the upcoming online healthcare website project using the Function Point Analysis method, you can follow these steps:

1. Identify the five function point components:
   - External Inputs (EI)
   - External Outputs (EO)
   - External Inquiries (EQ)
   - Internal Logical Files (ILF)
   - External Interface Files (EIF)
2. Calculate the unadjusted function points (UFP) for each component:
   - For External Inputs (EI):
     - Simple Inputs: 20
     - Complex Inputs: 20
     - UFP for EI = (Simple Inputs × 3) + (Complex Inputs × 4) = (20 × 3) + (20 × 4) = 60 + 80 = 140
   - For External Outputs (EO):
     - All Outputs are complex: 20
     - UFP for EO = (Complex Outputs × 5) = (20 × 5) = 100
   - For External Inquiries (EQ):
     - Simple Inquiries: 10
     - Complex Inquiries: 10
     - UFP for EQ = (Simple Inquiries × 3) + (Complex Inquiries × 4) = (10 × 3) + (10 × 4) = 30 + 40 = 70
   - For Internal Logical Files (ILF):
     - All data structures/logical files are complex: 1 (Patient File + Medicine Details File + Order Details File)
     - UFP for ILF = (Number of ILFs × 7) = (1 × 7) = 7
   - For External Interface Files (EIF):
     - Both interfaces are simple: 2
     - UFP for EIF = (Number of EIFs × 5) = (2 × 5) = 10
3. Calculate the total unadjusted function points (UFP): Total UFP = UFP for EI + UFP for EO + UFP for EQ + UFP for ILF + UFP for EIF Total UFP = 140 + 100 + 70 + 7 + 10 = 327
4. Calculate the adjusted function points (AFP) based on the degree of influence (DI):

AFP = Total UFP × (0.65 + 0.01 × DI) AFP = 327 × (0.65 + 0.01 × 42) = 327 × (0.65 + 0.42) = 327 × 1.07 = 349.89

Rounded to the nearest whole number, the adjusted function points (AFP) for the project is approximately 350.

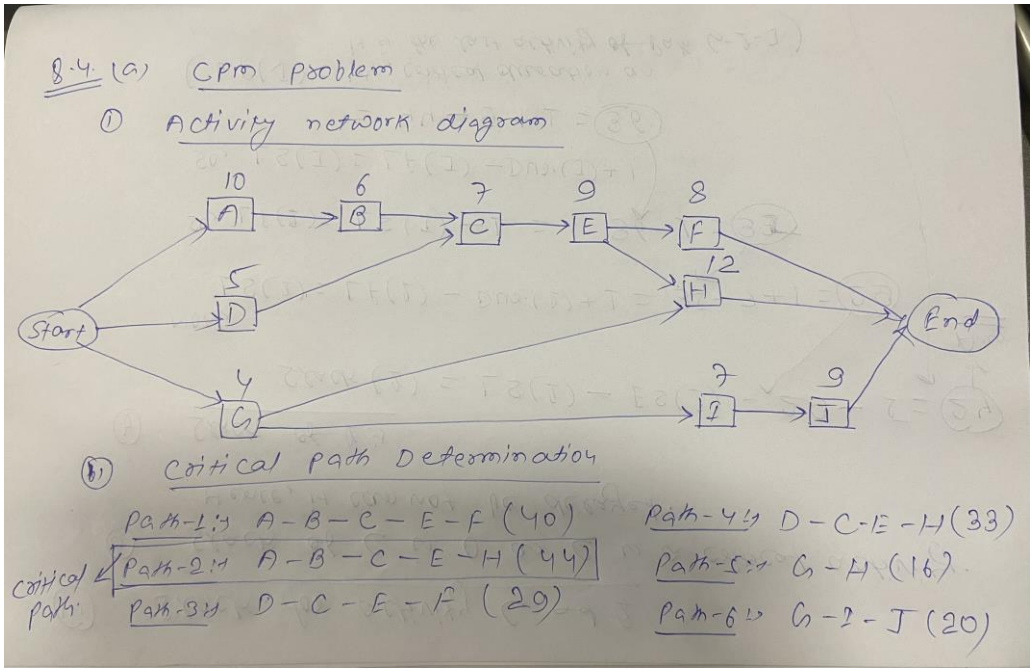
4. (a). Consider a software project development scenario that consists of the activities namely A, B, C, D, E, F, G, H, I, and J. The duration, in weeks, of the activities is 10, 6, 7, 5, 9, 8, 4, 12, 7, and 9 respectively. While doing the analysis, the team observed that activities A, D, and G can be started in parallel. Activity B can be started after the completion of activity A. Again, activities C can be started only when B and D gets over. Activities E can start only if the activity C is complete. After the completion of activity E, the team can start activity F and H. When activity G is complete the team can initiate the work on activities H and I. After the completion of activity I, activity J can be initiated. Do the following for the above discussed scenario:

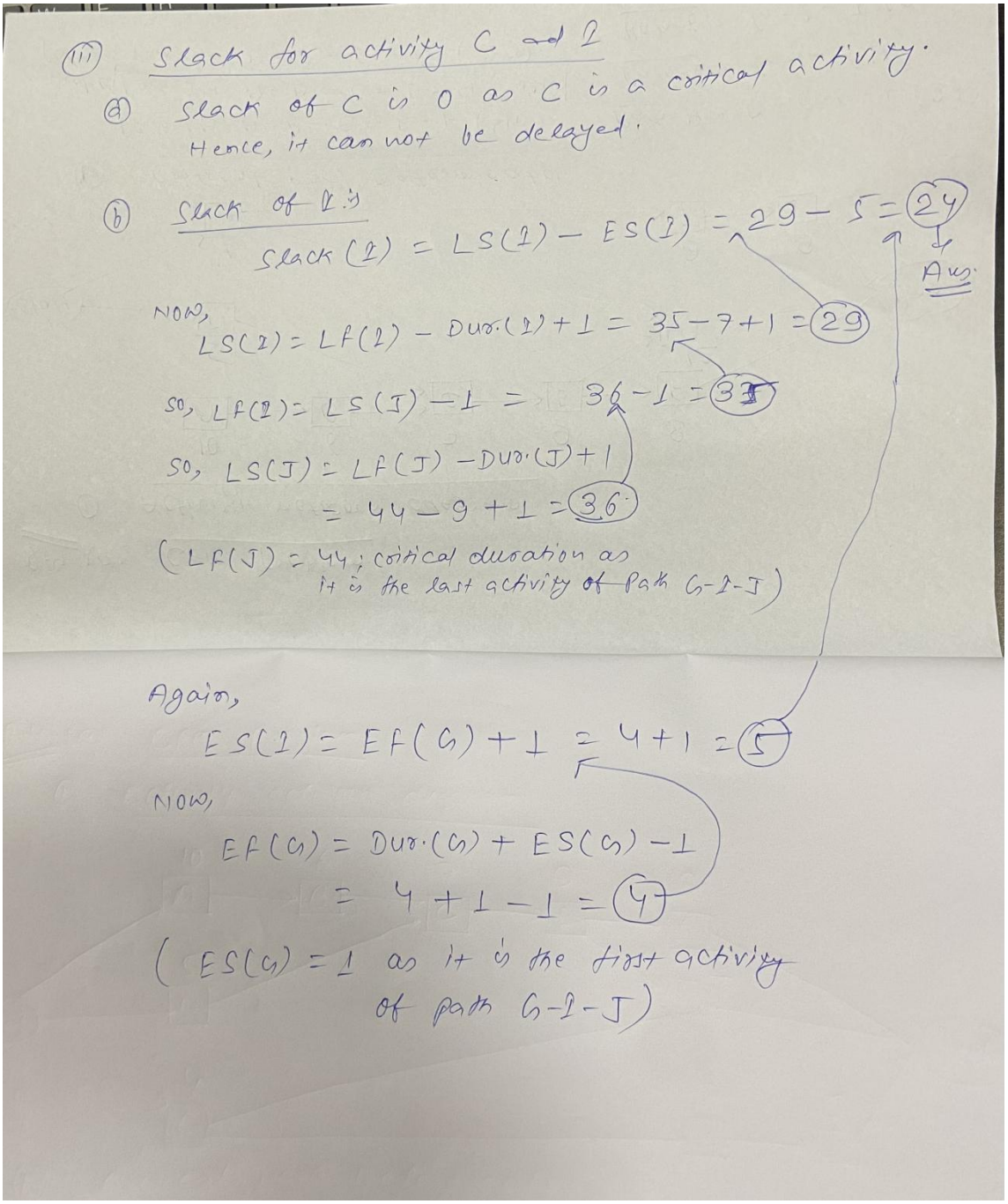
I. Draw the activity network diagram.

II. Determine the critical path.

III. Calculate the slack for activity C and I.

[5 Marks]





(iii) Slack for activity C and I

(a) Slack of C is 0 as C is a critical activity.
Hence, it can not be delayed.

(b) Slack of I is

Slack (I) = LS(I) − ES(I) = 29 − 5 = 24

Ans.

NOW,

LS(I) = LF(I) − Dur.(I) + 1 = 35 − 7 + 1 = 29

So, LF(I) = LS(J) − 1 = 36 − 1 = 35

So, LS(J) = LF(J) − Dur.(J) + 1

= 44 − 9 + 1 = 36

(LF(J) = 44; critical duration as it is the last activity of Path G-I-J)

Again,

ES(I) = EF(G) + 1 = 4 + 1 = 5

NOW,

EF(G) = Dur.(G) + ES(G) − 1

= 4 + 1 − 1 = 4

(ES(G) = 1 as it is the first activity of path G-I-J)

(b). What are the appropriate circumstances for using the RAD model? Discuss the strength and weakness of the RAD model. Also, compare the RAD model with the Spiral model. [5 Marks]

Appropriate circumstances for using the RAD model:

Project scope: focused, where the business objectives are well defined and narrow

Project data: already exist

Project decisions: can be made by a small number of people who are available and co-located

Project team: very small (six people)

Project technical architecture: defined and clear and the key technology components are well known

Project Technical requirements: are reasonable and well within the capability of the technology being used.

RAD Strengths:

- Reduced cycle time and improved productivity with fewer people means lower costs
- Time-box approach mitigates cost and schedule risk.
- Customer involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs
- Focus moves from documentation to code.
- Uses modeling concepts to capture information about business, data, and processes.

RAD Weaknesses:

- Accelerated development process must give quick responses to the user
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

S.No.	RAD MODEL	SPIRAL MODEL
1.	RAD model is a software development model where by the components or functions are developed in parallel as if they were mini projects.	Spiral model is a software development model and is made with features of incremental, waterfall or evolutionary prototyping models.
2.	Rapid development is its main objective.	High assurance is its main objective.
3.	RAD model requirements and early stage planning is not necessary.	Spiral model requirements and early stage planning is required.
4.	It is necessary to have detailed documentation but in a limited manner.	Detailed documentation is required.
5.	Requirements are specified as time boxed release manner.	Requirements are specified in the beginning.
6.	RAD model is used between large and small project.	Spiral model is used for large project.
7.	There is low amount risk in RAD model.	There is medium to high amount risk in spiral model.
8.	In RAD model small team size is required.	In spiral model large team is required.
9.	Flexibility to change in RAD model is Easy.	Flexibility to change in spiral model is not that difficult.
10.	In RAD model overlapping of phases is possible.	In spiral model overlapping of phases is not possible.
11.	Testing is done in RAD model after completion of coding.	Testing is done in spiral model at the end of the engineering phase.
12.	Cost of RAD model is Low.	Cost of spiral model is very expensive.
13.	Customer involvement is only at the beginning.	Customer involvement is high as compared to RAD model.
14.	RAD model is having easy maintenance.	Spiral model has typical maintenance.
15.	It offers reusability.	Reusability is possible to some extent.

5 (a). Assume that the development of a software project requires a large number of team members ,and a lot of innovations. The requirements are not stable and hence, lots of risks are associated with the project. The size of the project has been estimated to be 20,100 source lines of code. Assume that the average salary of software engineers be Rs. 40,000/- per month. Calculate the following:

I. Effort required to develop the software product
II. Development time
III. Productivity
IV. Average staff
V. Cost to develop the project

[5 Marks]

There seems to be a two-front answer to the above questions, as there exists two different scenarios:

(i) first on the basis of the number of lines of code, and (ii) the second on the basis of keywords like " large number of team

members", " lot of innovations", and " lots of risks are associated". In view of the above scenarios:

(i) the first case can be classified into organic Basic COCOMO (in between 2-50 KLOC, in question we have 20.1 KLOC)

(ii) for the second case, we land into the embedded Basic COCOMO.

Accordingly, we calculate the values based on both the cases.

(i) For Organic Basic COCOMO:

KLOC = 20.1

Effort required to develop the software product = $2.4\ (KLOC)^{1.05}$PM = 2.4(20.1)1.05= 56.04 PM

Development time = $2.5\ (Effort)^{0.38}$ Months= 2.5 (56.04)0.38 Months= 11.54 Months

Productivity = (Size / Effort) = (20.1/56.04) KLOC/PM = 0.3586 KLOC/PM

Average Staff = (Effort / Development Time)= 56.04 / 11.54= ⌈ 4.856⌉ = 5 Persons

Cost to develop the project = Effort x average salary per month= 56.04 x 40,000= 22,41,600 /-

(ii) For Embedded Basic COCOMO:

KLOC = 20.1

Effort required to develop the software product = 3.6 $(KLOC)^{1.20}$PM= 3.6(20.1)1.20= 131.86 PM

Development time = 2.5 $(Effort)^{0.32}$ Months= 2.5 (131.86)0.32 Months= 11.92 Months

Productivity = (Size / Effort) = (20.1/131.86) KLOC/PM = 0.1524 KLOC/PM

Average Staff = (Effort / Development Time= 131.86 / 11.92= ⌈ 11.06⌉ = 12 Persons

Cost to develop the project = Effort x average salary per month= 131.86 x 40,000= 52,74,400 /-

(b). What do you understand by the principles of abstraction and decomposition? Why are these two principles considered important in software engineering? Explain the problems that these two principles target to solve? Support your answer using suitable examples. [5 Marks]

Abstraction:

It refers to the construction of a simpler version of a problem by ignoring the details. The principle of constructing an abstraction is popularly known as modeling .

It is the simplification of a problem by focusing on only one aspect of the problem while omitting all other aspects. When using the principle of abstraction to understand a complex problem, we focus our attention on only one or two specific aspects of the problem and ignore the rest.

Whenever we omit some details of a problem to construct an abstraction, we construct a model of the problem. In everyday life, we use the principle of abstraction frequently to understand a problem or to assess a situation.

Decomposition:

Decomposition is a process of breaking down. It will be breaking down functions into smaller parts. It is another important principle of software engineering to handle problem complexity. This principle is profusely made use of by several software engineering techniques to contain the exponential growth of the perceived problem complexity. The decomposition principle is popularly is says the divide and conquer principle.

Functional Decomposition:

It is a term that engineers use to describe a set of steps in which they break down the overall function of a device, system, or process into its smaller parts.

Steps for the Functional Decomposition:

1. Find the most general function

2. Find the closest sub-functions

3. Find the next levels of sub-functions

*** Best of Luck ***