NAME: RICHA PAGARE
STUDENT ID: 1001873138
HOMEWORK 03

1) **Why should you be interested in learning about Lambda Calculus?**

Ans: We should be interested in learning about Lambda Calculus due to the following reasons:

- Lambda Calculus can encode any computation. It is a core model of computation. Any programming language can be encoded in Lambda Calculus. It is Turing-complete and can represent almost any kind of computation using clever encoding. Ideas of any computation can be developed and encoded in lambda calculus, and then scaled to full languages.
- Serves as a model for many functional languages, like for example Haskell, and Machine Learning. Hence, it is safe to say that lambda calculus serves as a basis model for almost all of functional languages which have later evolved. It is useful for understanding how languages work.
- It is now present in most of the major programming languages, which may or may not be functional. For example, Java, C# are not functional. However, they encode lambda calculus.

2) **How do you encode the concepts of TRUE, FALSE, NOT, AND, OR?**

Ans:

a) **TRUE:** We can encode TRUE as:
λx. λy. x

b) **FALSE:** We can encode FALSE as:
λx. λy. y

c) **NOT:** This can be encoded as: λa. a FALSE TRUE.
For example, if the input is true (a =TRUE), then it will be
(λa a FALSE TRUE) TRUE
=> (λa.a False True)True
=>(λa.a False True)(λx.λy.x)
=>(λx.λy.x) False True
=>False

This gives FALSE as output.

d) **AND**: It can be encoded as: λa. λb. a b a.

For example,
if a = FALSE and b = TRUE
(λa λb a b a) FALSE TRUE
=> FALSE TRUE FALSE
This gives FALSE as output.

e) **OR:** This can be encoded as:
λa. λb a a b

3) What is important about the Lambda Calculus expression called 'Y Combinator'?

Ans: The Y combinator allows to perform recursion in a language which does not support recursion. Y combinator is used in order to define a recursion in Lambda calculus.

4) Write the Y Combinator expression in Lambda Calculus.

Ans: The Y Combinator expression is as follows

Y = λf. (λx. f(x x)) (λx f(x x))

5) Where did the language 'Haskell' get its name?

Ans: The language Haskell gets its name after Haskell Curry. Haskell Curry was actually an early logician who worked with Lambda Calculus. Thus, the language got its name after him.

6) In the video it was mentioned that Erlang was used to code what?

Ans: Erlang was used to code the famous messaging app 'Whatsapp'.

7) How is 'pattern matching' used?

Ans: A function can have multiple definitions associated with it based on what value it is given as input. In a nutshell, pattern matching is used to identify their respective function definition on these different kinds of use cases of inputs. For example, a search function will return nothing in case of empty input array, but it will perform a search on an array with the given value when program is defined using the scenario based on type of input.

8) Complete this sentence: "NP problems are hard to solve but easy to _____"?

Ans: NP problems are hard to solve but easy to check.

9) What is the example of an NP problem used in the video?

Ans: Example of an NP problem used in the video is factoring a huge number.

10) What are the TV shows mentioned in the video?

Ans: 'The Simpsons' and 'Futurama' were mentioned in the video.

11) Floating point numbers are essentially what?

Ans: These are essentially scientific notations. For example, 13500000000 can be written as 1.35×10^{10} and so on.

12) Computers perform scientific notation in what base?

Ans: The computers perform scientific notations in base 2.

13) What is the problem with adding 1/3 + 1/3 + 1/3 using base 10 and ignoring recurring numbers?

Ans: The problem with this is that we would get floating point rounding errors. If we take the sum of 1/3 + 1/3 + 1/3, it comes out to be 0.999999… which we approximate to be 1. But, if we ignore the recurring bits the sum would be
0.3 + 0.3 + 0.3 that is 0.90, which would result in an error.

14) What is 1/10 in base 2?

Ans: It will be **0.0001100110011…**

15) What is the name of the function discussed in the video?

Ans: Ackermann's function was discussed in the video.

16) Can Ackermann's function be coded using for or 'DO' loops?

Ans: Ackermann's function cannot be coded using for or 'DO' loops. Ackermann's function is not a primitive recursive function, i.e. it cannot be defined by loops.

17) What is the value of Ackermann (4,1)?

Ans. Ackermann (4, 1) is 65533.

18) How many minutes will the machine in the video take to calculate Ackermann (4,2).

Ans In order to calculate Ackermann (4 ,2), it would actually require lots of time. It would take (2^{65533} x 3) minutes.

19) The performance characteristic of Ackermann's function is described as what?

Ans: Super Exponential

20) A loop nested in another loop has the performance characteristic of what?

Ans: Nested loops have performance characteristics of multi-dimensional problem, and problem with nested loops is n squared problem.

21) What was the limitation of Fortran mentioned in the video?

Ans: The major limitation of FORTRAN mentioned is that it did not do user level recursion. Also, early Fortran compilers didn't allow nested loops to go more than 10 deep.

22) What real-world use needs complex recursion?

Ans: Compilers. The compilers need complex recursion in order to check nesting of brackets in the program in the program syntax, etc.

23) There was a need to have a language that could cope with what?

Ans: There was a need to have a language that could cope with different widths of objects. There were several computers with several widths and the language needed to cope with all these different widths.

24) C is most powerful when considered as the classical what?

Ans: It is the most powerful when considered as the classical System Implementation Language**.** In earlier one of the hardest challenges was to implement an Operating System for the computer, as it had to be done in an assembler. This is when C language came into play.

25) What are the names of the two fields of the 'THING' structure?

Ans: The two fields are**:**

- **a.** char * item This is the pointer to a char, or the content in that linked list. It points to the first character in a string, which basically means that we are pointing at the whole string.
- **b.** struct_thing *next: This points to the next THING along in our list.

26) What is the advantage of the 'Triple Ref Technique'?

Ans: The Triple Ref Pointers are pointers pointing to pointers. Triple pointer is a pointer that point to another pointer, which in turn points to another pointer. The main advantage of the Triple Ref technique is that one does not have to keep a track of the previous node's location.

It can be used when we need to insert a new node into the linked list at a specified position. But, in the Triple Ref Technique, we make use of a tracer, which is a Triple ref pointer. We can say that a tracer is basically used to point to objects temporarily. This tracer, first points to the head and checks for the value of the next node from its current node. By inspecting at the next node's value, it decides if it has to move to next node, or to insert new node after the current node.

27) What is the procedure used in the video to compare the different structures?

Ans: In the video, the following procedure is used:

a. A struct of an array and a linked list both are made. The only difference is that linked list has an extra item which is the next pointer to next thing.
b. A random number generator is used to allocate 125,000 random numbers to each.
c. Then, we calculate the sum of every element of linked list. We do the same for array and calculate the sum.
d. Finally, we then record the time that function takes to calculate this sum in terms of clock ticks. We compare the times taken by the list and array to compare the different structures.

28) Why is the reverse array faster on the Atari?

Ans: The reverse array is faster on Atari as on an instruction level. Atari favors going backwards and makes use of somewhat different instructions. As a result, the code runs a little faster.

29) What would be the goal of requiring people to be exposed to coding?

Ans: People exposed to coding would understand the functioning of a computer better and in turn start thinking computationally. If people are exposed to coding, they need to understand how a computer system works and what goes around in the systems to do things the way they do things. It will also develop an appreciation of how things work. It allows people to think logically and young children should be exposed so that it opens their mind and they know how to tackle a problem.

30) List 3 or more of the different sort algorithms mentioned in the video.

Ans: The following Sorting Algorithms are mentioned in the video:

- Bubble Sort
- Quick Sort
- Selection Sort
- Heap Sort

31) What is the 'Decision Problem'?

Ans: Decision Problem is a test to determine whether a certain formula/function gives a certain answer or not. In the decision problem, we want to know if there is an automatic way of finding if the given premises or assumptions would lead to a conclusion or not.

32) An example of an abstraction used in the video is, "A transistor is a type of ___"?

Ans: To explain abstraction, it is said that a transistor is a type of switch, that could be turned opened or close.
Abstraction is the hiding of background details and showing only the essential information.

33) Which video was the most interesting or your favorite?

Ans: The video I found most interesting was Triple Ref pointers. I always found the concept difficult to understand. The magic trick of pointers to pointers which seemed very complicated to me actually simplified the code. I ended up implementing it and now have a solid understanding of how that works! It was a point explanation of insertion into linked lists.