

**संगणकशास्त्र विभाग**

न्यायमूर्ती रानडे भवन, तळ मजला,  
विद्यानगरी, कालीना, सांताक्रुझ (पूर्व),  
मुंबई - ४०० ०९८. (भारत)  
दूरध्वनी क्र. : २६५४ ३४४६ (विभाग प्रमुख)  
२६५४ ३३११ (कार्यालय)  
२६५२ ८८७४ (थेट)

University of Mumbai



मुंबई विद्यापीठ

**DEPARTMENT OF COMPUTER SCIENCE**

Justice Ranade Bhavan, Ground Floor,  
Vidyanagari, Kalina, Santacruz (East),  
Mumbai — 400 098. (INDIA)  
Tel. No. : 2654 3446 (HOD)  
2654 3311 (Office),  
2652 8874 (Direct)

E-mail : head@udcs.mu.ac.in

Website : www.mu.ac.in/science/cs/profile.html

**Ref. No. UDCS/ 62 / 2013**

**Date: 6<sup>th</sup> February, 2013.**

Hello

I-Know-Vention 2013 is on and we are happy to put it on the record that 12000 students have registered and most of them have taken the scrutiny test.

The next round of elementary programming contest received a good response.

We are ready with the everyday quiz through FaceBook...

Remember the excitement that you enjoyed last year.

With the same zeal and enthusiasm you will play the on-line quiz this year too

It starts today. Visit [www.seed-iknowvention.com](http://www.seed-iknowvention.com)

Teach C contest of Steve Jobs track has been announced.

Also please find the details of your questions posted at [www.placementgym.blogspot.com](http://www.placementgym.blogspot.com)

Start build up teams, pick up a topic, work out a plan and create e-contents for teaching C using Spoken Tutorials and Text book companion.

Last date of submission is 17th Feb. Steve Jobs trophy is waiting for you...

List of practice problems for the Tech C DMR programming contest has been displayed.

Please go through the same, who knows, you will be the winner of the trophy if you prepared and come prepared, attempt the programming contest...

Hurry up!

Thanks and all the best!

(Dr. Ambuja R. Salgaonkar)

Head

Department of Computer Science

# “Teach C Steve Jobs track of I-Know-Vention

Objectives:

(Long term) Teaching C through a good book and by using a sophisticated platform.

(Short term) Learning C from “C programming language” by K & R and learning Ch-platform

(Immediate) Developing Spoken tutorials on the topics in K&R and text book companion by using Ch / GNU platform

What you have to do:

<http://www.softintegration.com/download/>

Download students’ version by stating that you will use it for learning-teaching purpose. It is free (as in freedom). Understand the sophistication in the IDE that will help acquire C skills.

Get “C programming language” by Kernighan and Ritchie (K&R). Its not so costly, also available at various sites from where could be downloaded at no cost. Answer book is the solution of the same.

Example site: <http://anshulmalik.net/ebooks/download-the-c-programming-language-pdf/>

Learn what is Spoken Tutorial and how to create it from the site: <http://spoken-tutorial.org/>

See Spoken Tutorial [http://www.spoken-tutorial.org/script/index.php/C with GCC](http://www.spoken-tutorial.org/script/index.php/C_with_GCC)

Follow the methodology and create spoken tutorial on any one of the topics in the following list.

The topics will be allotted on the first come basis. At the most two teams could share a topic and create a spoken tutorial on the same independently. The team leaders (teacher-coordinator of the team) are requested to send an email to [ambujas@gmail.com](mailto:ambujas@gmail.com) to confirm the availability of the topic.

The team also has to solve the exercises on the same topic that are given in the K&R book by using this new platform.

The spoken tutorial + the solution of the exercises together will form your submission that will be considered for the evaluation while announcing the winner of the Steve Jobs trophy on 18<sup>th</sup> February 2013. Submission is possible on or before 16<sup>th</sup> February.

The marks obtained for submitting the Spoken Tutorial on C using the sophisticated platform provided by Softintegration would carry weight in counting the score of the general championship of I-Know-Vention 2013

Thanks and all the best for e-content-creation and for winning SJ trophy...

List of topics:

1. What is programming? Why should we learn C? *Do not list out the merits and demerits; Prepare some catchy and informative live demo, clip etc., that should speak for itself; Tell the learner from where to download and how to install C on different platforms.*

2. Introduce the terms: source code, compilation, run, syntax and semantics, errors and how to remove them on windows and Unix / Linux platforms; Give illustrations and create interesting assignments that would speak for the errors.
3. A C program: *Skeleton of a C program needs to be explained; This is a place where the power of programming can be introduced; Create some interesting real life illustrations and assignments that involve only display statements.*
4. Introduction to discrete variables and the assignment operator: *Explain the concepts with the help of the entities they know in the real world; Next, introduce arithmetic operators, increment and decrement operators; Illustrate some widely known difficult computations and demonstrate the accuracy of the results when programmed; Generate a list of such assignments; Also, explain the capacity of a variable to hold the data and show how it is dependent of the platforms (hardware and software).*
5. Assignment, equal and equivalent: *Meaning of the terms, interesting illustrations, availability in C and other languages, introduce a tool for practicing these concepts*
6. Comments and declarations: *What are they in the real world? What is their role in programming? What are the ways that they are available in the programming world? Bring the range to the notice of a learner and illustrate how it caters the diverse applications' requirements; create some thought provoking questions to test the understanding of the students after taking this tutorial.*
7. Concrete Vs. Abstract data types: *Create Audio-visual demos to explain these two concepts; provide a practice session on the data types in C.*
8. Declarative Vs. Imperative programs: *Create illustrations and explain how the declarative and imperative computers work. Explain why C is an imperative programming language.*
9. Functions: *The dictionary word, interpretation in real world and interpretation in Mathematics, signature of a function, functions in C, and functions in other languages; 'main' is a function; necessity of main, necessity of other functions, introduction to built-in functions and header files; Interesting demos only by using the built-in functions and, a list of assignments could be created.*
10. macros and pre-processing: *Dictionary meaning of the terms, examples of real life usage of the terms, examples in programming world, the case of C programming, introduction to #define and tool to practice this concept; difference between functions and macros.*
11. Relational operators: *Create an audio-visual aid to explain the concept of ordering; the definition is dependent of what you perceive as an order. The learner may select the definition and see the effect, Better, if the tool facilitates the learner to construct her own definitions of order and see the effect; Introduce the concept of relational operators in this context, introduce the relational operators in C, Create an interactive practice session for developing understanding about all the relational operators in C.*
12. Operator precedence: *Illustrate its importance and create practice examples*
13. User defined functions: *Monolithic Vs. Modular approach in general, prepare a demo of real world applications of these terms; show differently how would you make pupils appreciate the*

*modular over monolithic; Come to the programming world, develop an understanding about the need of modular approach of programming, take examples of some fascinating projects and provide visualization of them in parts, Discuss the range of modularity available; Come to the C programming, show how to write your own functions and use them in a program, save them in a header file and share the definitions in diverse applications. Create a list of interesting assignments (should not involve complex data structures and difficult logic, see to that they involve enough non-trivial and catchy topics)*

14. Program memory organization: *Explain the existence of stack and heap in real (non-programming) world, Introduce the terms data segment, stack segment and code segment in the programming context; show how the memory allocation takes place in C.*

15. Scope and life of the variables: *What are declaration, definition and Initialization; introduce the terms compile time and run time; illustrate local, global, auto, static and extern variables; Show how to trace their existence in memory and hence comment on their behaviour; Comment on the pros and cons of using each of them.*

16. Iterative structures: *What is iteration? Find out some funny ways of introducing what it meant by doing a thing again and again. Why do we do so? Why not a human being? Why computers should be employed for these jobs? Select examples, demos to make it appealing! Introduce how a C program can be written to instruct a machine to do a particular activity repeatedly; what are the other ways in C to do the similar things? Give some assignments that involves iterations and they are sensible tasks in the real life.*

17. Variety of iterative structures in C: *Select appealing real life examples to illustrate when to prefer which structure; Develop a game to train the students to select an appropriate iterative structure in a given a situation / problem; System should provide feedback to the learners and help them improve their understanding about the functioning of the different iterative structures.*

18. Logical operators: *What is a logic system? N-valued logic and value of an expression; Boolean as a special case, i.e., 2-valued logic and truth value of a statement; Functions Vs. Predicates, create an interactive practice session to play with these concepts; Discuss the semantics of predicates AND, OR, NOT, Ex-OR and their corresponding syntax in C, Comparison with the semantics of these words in English; Shift operators; Develop an interactive tool that allows the user to construct a Boolean expression and computes its value, conversely, the tool will generate a Boolean expression and verify the user's response.*

19. Selection structures: *Selection structure and decision point, real life examples – maybe cartoon stories to explain how many times we come across the decision points and how many are the possible ways one can see to resolve the issues; Generating a decision tree to model different decision paths of different pay-offs; Discuss different ways of modelling the decision points in C; Develop an interactive tool that displays a tree as per the given specifications and generates an equivalent C statement to represent the portion that the user selects. Conversely, given a C statement, the tool may display the equivalent decision tree; Comparison of else with logical OR.*

20. Indenting and bracketing: *Semantics of white spaces, flowery brackets, rectangular brackets and round brackets in C and some other languages; beautification for readability and readability for maintenance; standard good programming practices; You may conduct interviews of the professional programmers, gather data from the experiences that have been shared on the social networking sites, read project manuals, generate inputs on your own.*

21. Input / Output: `textstream... #include<stdio.h>`

22. Formatted input-output: Impressive illustrations and realistic exercises

23. Escape sequences: Illustrations and exercises

24. Integer and floating point arithmetic: Comparison, applications of the both, illustrations and exercises

25. Implicit and explicit type conversions: *Discuss typed Vs. Un-typed languages, strong Vs. Weak type-checked languages, static Vs. Dynamic type-checked languages; type-checking and safety; Develop a tool to classify the language depending upon the error pattern in mixed data-type operations and vice-versa i.e., given the class of a language and an expression involving operands of the mixed data types, generates an error if any; the type checking and conversion mechanism in C; flexibility and concerns.*

26. Infinite loop, break and continue: Illustration, applications and exercises

27. while and for: *Comparison and applications, conversions; exercises on the same*

28. Files: File interface, illustration, application and exercises

29. String processing: `#include <string.h>`

30. Lists / Arrays: Illustrations, applications and exercises

31. Pointers: Illustrations, applications and exercises

32. Memory models: Tiny, small, medium, compact, large and huge

33. Call by value and call by reference: Comparison, illustrations and exercises

34. Sorting-1: *Develop a tool that accepts the data from user, also accepts the definition of order and show graphically the step-by-step process of sorting using brute-force or generate-and-test approach for sorting; While sorting, the tool must show the number of comparisons and assignments that took place in the process; It must have ability to take small and big datasets and sort them; Its good if the tool generates a graph of 'the data size' against 'the number of computations involved' in sorting; Better if the tool also explains / interprets the characteristic.*

35. Sorting-2: *Develop a tool that accepts the data from user, also accepts the definition of order and show graphically the step-by-step process of sorting using Bubble-sort algorithm for sorting; While sorting, the tool must show the number of comparisons and assignments that took place in the process. It must have ability to take small and big datasets and sort them; Its good if the tool*

*generates a graph of 'the data size' against 'the number of computations involved' in sorting; Better if the tool also explains / interprets the characteristic.*

36. Sorting-3: Develop a tool that accepts the data from user, also accepts the definition of order and show graphically the step-by-step process of sorting using Quick-sort algorithm for sorting; While sorting, the tool must show the number of comparisons and assignments that took place in the process; It must have ability to take small and big datasets and sort them; Its good if the tool generates a graph of 'the data size' against 'the number of computations involved' in sorting; Better if the tool also explains / interprets the characteristic.

37. Searching: Develop a tool that accepts the data from user, also accepts the key and show graphically the step-by-step process of linear search; While searching, the tool must show the number of comparisons and assignments that took place in the process; It must have ability to take small and big datasets; Its good if the tool generates a graph of 'the data size' against 'the number of computations involved' in searching; Better if the tool also explains / interprets the characteristic.

38. Searching: Develop a tool that accepts the sorted data from user, also accepts the key and show graphically the step-by-step process of n-ary search; While searching, the tool must show the number of comparisons and assignments that took place in the process; It must have ability to take small and big datasets; Its good if the tool generates a graph of 'the data size' against 'the number of computations involved' in searching; Better if the tool also explains / interprets the characteristic.

39. Merging: Develop a tool that accepts the sorted files from user, shows graphically the step-by-step process of merging; While merging, the tool must show the number of comparisons and assignments that took place in the process; It must have ability to take small and big datasets; Its good if the tool generates a graph of 'the data size' against 'the number of computations involved' in searching; Better if the tool also explains / interprets the characteristic.

40. Hashing: Develop a tool that accepts the data, hash-function and a key from user, shows graphically the step-by-step process of creating and look-up in hash and hash-search; While doing so, the tool must show the number of comparisons and assignments that took place in the process; It must have ability to take small and big datasets; Its good if the tool generates a graph of 'the data size' against 'the number of computations involved' in the process; Better if the tool also explains / interprets the characteristic

41. Hashing: Develop a tool that accepts the data, hash-function and a key from user, shows graphically the step-by-step process of creating and look-up in hash and hash-search; While doing so, the tool must show the number of comparisons and assignments that took place in the process; It must have ability to take small and big datasets; Its good if the tool generates a graph of 'the data size' against 'the number of computations involved' in the process; Better if the tool also explains / interprets the characteristic

42. Time complexity of an algorithm: ???

43. Scope of the variable: behaviour of local, global, static, auto, extern variables with the reasoning, pros and cons. A variable can be defined more than once as long as the declarations are consistent. Variable cannot be defined many times (it produces linking error). Extern is explicit declaration to

mention that the definition is elsewhere. Initialization is explicit declaration with the definition at the same place.

44. Logical operators: *What is a logic system? N-valued logic and value of an expression; Boolean as a special case, i.e., 2-valued logic and truth value of a statement; Functions Vs. Predicates, create an interactive practice session to play with these concepts; Discuss the semantics of predicates AND, OR, NOT, Ex-OR and their corresponding syntax in C, Comparison with the semantics of these words in English; Shift operators; Develop an interactive tool that allows the user to construct a Boolean expression and computes its value, conversely, the tool will generate a Boolean expression and verify the user's response.*

45. Selection structures: *Selection structure and decision point, real life examples – maybe cartoon stories to explain how many times we come across the decision points and how many are the possible ways one can see to resolve the issues; Generating a decision tree to model different decision paths of different pay-offs; Discuss different ways of modelling the decision points in C; Develop an interactive tool that displays a tree as per the given specifications and generates an equivalent C statement to represent the portion that the user selects. Conversely, given a C statement, the tool may display the equivalent decision tree; Comparison of else with logical OR.*