Algorithms and Programming

1. Given a rectangular (cuboidal for the puritans) cake with a rectangular piece removed (any size or orientation),how would you cut the remainder of the cake into two equal halves with one straight cut of a knife?

2. You're given an array containing both positive and negative integers and required to find the sub-array with the largest sum (O(N) a la KBL). Write a routine in C for the above.

3. Given an array of size N in which every number is between 1 and N, determine if there are any duplicates in it. You are allowed to destroy the array if you like. [I ended up giving about 4 or 5 different solutions for this, each supposedly better than the others].

4. Write a routine to draw a circle (x ** 2 + y ** 2 = r ** 2) without making use of any floating point computations at all. [This one had me stuck for quite some time and I first gave a solution that did have floating point computations].

5. Given only putchar (no sprintf, itoa, etc.) write a routine putlong that prints out an unsigned long in decimal. [I gave the obvious solution of taking % 10 and / 10, which gives us the decimal value in reverse order. This requires an array since we need to print it out in the correct order. The interviewer wasn't too pleased and asked me to give a solution which didn't need the array].

6. Give a one-line C expression to test whether a number is a power of 2. [No loops allowed - it's a simple test.]

7. Given an array of characters which form a sentence of words, give an efficient algorithm to reverse the order of the words (not characters) in it.

8. How many points are there on the globe where by walking one mile south, one mile east and one mile north you reach the place where you started.

9. Give a very good method to count the number of ones in a "n" (e.g. 32) bit number.

ANS. Given below are simple solutions, find a solution that does it in log (n) steps.

Iterative

```
function iterativecount (unsigned int n)
begin
  int count=0;
```

```
 while (n)
 begin
   count += n & 0x1 ;
   n >>= 1;
 end
 return count;
end
```

Sparse Count

```
function sparsecount (unsigned int n)
begin
 int count=0;
 while (n)
 begin
   count++;
   n &= (n-1);
 end
 return count ;
end
```

10. What are the different ways to implement a condition where the value of x can be either a 0 or a 1. Apparently the if then else solution has a jump when written out in assembly. if (x == 0) y=a else y=b There is a logical, arithmetic and a data structure solution to the above problem.

11. Reverse a linked list.

12. Insert in a sorted list

13. In a X's and 0's game (i.e. TIC TAC TOE) if you write a program for this give a fast way to generate the moves by the computer. I mean this should be the fastest way possible.

The answer is that you need to store all possible configurations of the board and the move that is associated with that. Then it boils down to just accessing the right element and getting the corresponding move for it. Do some analysis and do some more optimization in storage since otherwise it becomes infeasible to get the required storage in a DOS machine.

14. I was given two lines of assembly code which found the absolute value of a number stored in two's complement form. I had to recognize what the code was doing. Pretty simple if you know some assembly and some fundaes on number representation.

15. Give a fast way to multiply a number by 7.

16. How would go about finding out where to find a book in a library. (You don't know how exactly the books are organized beforehand).

17. Linked list manipulation.

18. Tradeoff between time spent in testing a product and getting into the market first.

19. What to test for given that there isn't enough time to test everything you want to.

20. First some definitions for this problem: a) An ASCII character is one byte long and the most significant bit in the byte is always '0'. b) A Kanji character is two bytes long. The only characteristic of a Kanji character is that in its first byte the most significant bit is '1'.

Now you are given an array of a characters (both ASCII and Kanji) and, an index into the array. The index points to the start of some character. Now you need to write a function to do a backspace (i.e. delete the character before the given index).

21. Delete an element from a doubly linked list.

22. Write a function to find the depth of a binary tree.

23. Given two strings S1 and S2. Delete from S2 all those characters which occur in S1 also and finally create a clean S2 with the relevant characters deleted.

24. Assuming that locks are the only reason due to which deadlocks can occur in a system. What would be a foolproof method of avoiding deadlocks in the system.

25. Reverse a linked list.

Ans: Possible answers -

iterative loop
curr->next = prev;
prev = curr;
curr = next;
next = curr->next
endloop

```
recursive reverse(ptr)
if (ptr->next == NULL)
return ptr;
temp = reverse(ptr->next);
temp->next = ptr;
return ptr;
end
```

26. Write a small lexical analyzer - interviewer gave tokens. expressions like "a*b" etc.

27. Besides communication cost, what is the other source of inefficiency in RPC? (answer : context switches, excessive buffer copying). How can you optimize the communication? (ans : communicate through shared memory on same machine, bypassing the kernel _ A Univ. of Wash. thesis)

28. Write a routine that prints out a 2-D array in spiral order!

29. How is the readers-writers problem solved? - using semaphores/ada .. etc.

30. Ways of optimizing symbol table storage in compilers.

31. A walk-through through the symbol table functions, lookup() implementation etc. - The interviewer was on the Microsoft C team.

32. A version of the "There are three persons X Y Z, one of which always lies".. etc..

33. There are 3 ants at 3 corners of a triangle, they randomly start moving towards another corner.. what is the probability that they don't collide.

34. Write an efficient algorithm and C code to shuffle a pack of cards.. this one was a feedback process until we came up with one with no extra storage.

35. The if (x == 0) y = 0 etc..

36. Some more bitwise optimization at assembly level

37. Some general questions on Lex, Yacc etc.

38. Given an array t[100] which contains numbers between 1..99. Return the duplicated value. Try both O(n) and O(n-square).

39. Given an array of characters. How would you reverse it. ? How would you reverse it without using indexing in the array.

40. Given a sequence of characters. How will you convert the lower case characters to upper case characters. (Try using bit vector - solutions given in the C lib -typec.h)

41. Fundamentals of RPC.

42. Given a linked list which is sorted. How will u insert in sorted way.

43. Given a linked list How will you reverse it.

44. Give a good data structure for having n queues (n not fixed) in a finite memory segment. You can have some data-structure separate for each queue. Try to use at least 90% of the memory space.

45. Do a breadth first traversal of a tree.

46. Write code for reversing a linked list.

47. Write, efficient code for extracting unique elements from a sorted list of array. e.g. (1, 1, 3, 3, 3, 5, 5, 5, 9, 9, 9, 9) -> (1, 3, 5, 9).

48. Given an array of integers, find the contiguous sub-array with the largest sum.

ANS. Can be done in O(n) time and O(1) extra space. Scan array from 1 to n. Remember the best sub-array seen so far and the best sub-array ending in i.

49. Given an array of length N containing integers between 1 and N, determine if it contains any duplicates.

ANS. [Is there an O(n) time solution that uses only O(1) extra space and does not destroy the original array?]

50. Sort an array of size n containing integers between 1 and K, given a temporary scratch integer array of size K.

ANS. Compute cumulative counts of integers in the auxiliary array. Now scan the original array, rotating cycles! [Can someone word this more nicely?]

* 51. An array of size k contains integers between 1 and n. You are given an additional scratch array of size n. Compress the original array by removing duplicates in it. What if k << n?

ANS. Can be done in O(k) time i.e. without initializing the auxiliary array!

52. An array of integers. The sum of the array is known not to overflow an integer. Compute the sum. What if we know that integers are in 2's complement form?

ANS. If numbers are in 2's complement, an ordinary looking loop like for(i=total=0;i< n;total+=array[i++]); will do. No need to check for overflows!

53. An array of characters. Reverse the order of words in it.

ANS. Write a routine to reverse a character array. Now call it for the given array and for each word in it.

* 54. An array of integers of size n. Generate a random permutation of the array, given a function rand_n() that returns an integer between 1 and n, both inclusive, with equal probability. What is the expected time of your algorithm?

ANS. "Expected time" should ring a bell. To compute a random permutation, use the standard algorithm of scanning array from n downto 1, swapping i-th element with a uniformly random element <= i-th. To compute a uniformly random integer between 1 and k (k < n),call rand_n() repeatedly until it returns a value in the desired range.

55. An array of pointers to (very long) strings. Find pointers to the (lexicographically) smallest and largest strings.

ANS. Scan array in pairs. Remember largest-so-far and smallest-so-far. Compare the larger of the two strings in the current pair with largest-so-far to update it. And the smaller of the current pair with the smallest-so-far to update it. For a total of <= 3n/2 strcmp() calls. That's also the lower bound.

56. Write a program to remove duplicates from a sorted array.

ANS. int remove_duplicates(int * p, int size)
{

```
int current, insert = 1;
for (current=1; current < size; current++)
if (p[current] != p[insert-1])
{
p[insert] = p[current];
current++;
insert++;
}else
current++;

return insert;

}
```

57. C++ (what is virtual function ? what happens if an error occurs in constructor or destructor. Discussion on error handling, templates, unique features of C++. What is different in C++, (compare with unix).

58. Given a list of numbers (fixed list) Now given any other list, how can you efficiently find out if there is any element in the second list that is an element of the first list (fixed list).

59. Given 3 lines of assembly code : find it is doing. IT was to find absolute value.

60. If you are on a boat and you throw out a suitcase, Will the level of water increase.

61. Print an integer using only putchar. Try doing it without using extra storage.

62. Write C code for (a) deleting an element from a linked list (b) traversing a linked list

63. What are various problems unique to distributed databases

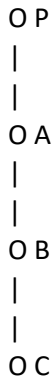
64. Declare a void pointer ANS. void *ptr;

65. Make the pointer aligned to a 4 byte boundary in a efficient manner ANS. Assign the pointer to a long number and the number with 11...1100 add 4 to the number

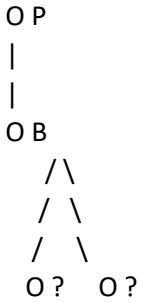
66. What is a far pointer (in DOS)

67. What is a balanced tree

68. Given a linked list with the following property node2 is left child of node1, if node2 < node1 else, it is the right child.



How do you convert the above linked list to the form without disturbing the property. Write C code for that.



determine where do A and C go

69. Describe the file system layout in the UNIX OS

ANS. describe boot block, super block, inodes and data layout

70. In UNIX, are the files allocated contiguous blocks of data

ANS. no, they might be fragmented

How is the fragmented data kept track of

ANS. Describe the direct blocks and indirect blocks in UNIX file system

71. Write an efficient C code for 'tr' program. 'tr' has two command line arguments. They both are strings of same length. tr reads an input file, replaces each character in the first string with the corresponding character in the second string. eg. 'tr abc xyz' replaces all 'a's by 'x's, 'b's by 'y's and so on. ANS.
a) have an array of length 26.
put 'x' in array element corr to 'a'
put 'y' in array element corr to 'b'

put 'z' in array element corr to 'c'
put 'd' in array element corr to 'd'
put 'e' in array element corr to 'e'
and so on.

the code
while (!eof)
{
c = getc();
putc(array[c - 'a']);
}

72. what is disk interleaving

73. why is disk interleaving adopted

74. given a new disk, how do you determine which interleaving is the best a) give 1000 read operations with each kind of interleaving determine the best interleaving from the statistics

75. draw the graph with performance on one axis and 'n' on another, where 'n' in the 'n' in n-way disk interleaving. (a tricky question, should be answered carefully)

76. I was a c++ code and was asked to find out the bug in that. The bug was that he declared an object locally in a function and tried to return the pointer to that object. Since the object is local to the function, it no more exists after returning from the function. The pointer, therefore, is invalid outside

.

77. A real life problem - A square picture is cut into 16 squares and they are shuffled. Write a program to rearrange the 16 squares to get the original big square.

78.
int *a;
char *c;
*(a) = 20;
*c = *a;
printf("%c",*c);

what is the output?

79. Write a program to find whether a given m/c is big-endian or little-endian!

80. What is a volatile variable?

81. What is the scope of a static function in C ?

82. What is the difference between "malloc" and "calloc"?

83. struct n { int data; struct n* next}node;
node *c,*t;
c->data = 10;
t->next = null;
*c = *t;
what is the effect of the last statement?

84. If you're familiar with the ? operator x ? y : z

you want to implement that in a function: int cond(int x, int y, int z); using only ~, !, ^, &, +, |, <<, >> no if statements, or loops or anything else, just those operators, and the function should correctly return y or z based on the value of x. You may use constants, but only 8 bit constants. You can cast all you want. You're not supposed to use extra variables, but in the end, it won't really matter, using vars just makes things cleaner. You should be able to reduce your solution to a single line in the end though that requires no extra vars.

85. You have an abstract computer, so just forget everything you know about computers, this one only does what I'm about to tell you it does. You can use as many variables as you need, there are no negative numbers, all numbers are integers. You do not know the size of the integers, they could be infinitely large, so you can't count on truncating at any point. There are NO comparisons allowed, no if statements or anything like that. There are only four operations you can do on a variable.
1) You can set a variable to 0.
2) You can set a variable = another variable.
3) You can increment a variable (only by 1),and it's a post increment.
4) You can loop. So, if you were to say loop(v1) and v1 = 10, your loop would execute 10 times, but the value in v1 wouldn't change so the first line in the loop can change value of v1 without changing the number of times you loop.

You need to do 3 things.
1) Write a function that decrements by 1.
2) Write a function that subtracts one variable from another.
3) Write a function that divides one variable by another.
4) See if you can implement all 3 using at most 4 variables. Meaning, you're not making function calls now, you're making macros. And at most you can have 4 variables. The restriction really only applies to divide, the other 2 are easy to do with 4 vars or less. Division on the other hand is dependent on the other 2 functions, so, if subtract requires 3 variables, then divide only has 1 variable left unchanged after a call to subtract. Basically,

just make your function calls to decrement and subtract so you pass your vars in by reference, and you can't declare any new variables in a function, what you pass in is all it gets.

Linked lists

* 86. Under what circumstances can one delete an element from a singly linked list in constant time?

ANS. If the list is circular and there are no references to the nodes in the list from anywhere else! Just copy the contents of the next node and delete the next node. If the list is not circular, we can delete any but the last node using this idea. In that case, mark the last node as dummy!

* 87. Given a singly linked list, determine whether it contains a loop or not.

ANS. (a) Start reversing the list. If you reach the head, gotcha! there is a loop!
But this changes the list. So, reverse the list again.
(b) Maintain two pointers, initially pointing to the head. Advance one of them one node at a time. And the other one, two nodes at a time. If the latter overtakes the former at any time, there is a loop!

```
p1 = p2 = head;

do {
    p1 = p1->next;
    p2 = p2->next->next;
}while (p1 != p2);
```

88. Given a singly linked list, print out its contents in reverse order. Can you do it without using any extra space?

ANS. Start reversing the list. Do this again, printing the contents.

89. Given a binary tree with nodes, print out the values in pre-order/in-order/post-order without using any extra space.

90. Reverse a singly linked list recursively. The function prototype is node * reverse (node *) ;

ANS.

```
node * reverse (node * n)
 {
   node * m ;

   if (! (n && n -> next))
     return n ;

   m = reverse (n -> next) ;
   n -> next -> next = n ;
   n -> next = NULL ;
```

```
    return m ;
  }
```

91. Given a singly linked list, find the middle of the list.

HINT. Use the single and double pointer jumping. Maintain two pointers, initially pointing to the head. Advance one of them one node at a time. And the other one, two nodes at a time. When the double reaches the end, the single is in the middle. This is not asymptotically faster but seems to take less steps than going through the list twice.

Bit-manipulation

92. Reverse the bits of an unsigned integer.

ANS.

```
#define reverse(x)                      \
        (x=x>>16|(0x0000ffff&x)<<16,        \
         x=(0xff00ff00&x)>>8|(0x00ff00ff&x)<<8, \
         x=(0xf0f0f0f0&x)>>4|(0x0f0f0f0f&x)<<4, \
         x=(0xcccccccc&x)>>2|(0x33333333&x)<<2, \
         x=(0xaaaaaaaa&x)>>1|(0x55555555&x)<<1)
```

* 93. Compute the number of ones in an unsigned integer.

ANS.

```
#define count_ones(x)                   \
        (x=(0xaaaaaaaa&x)>>1+(0x55555555&x),\
         x=(0xcccccccc&x)>>2+(0x33333333&x),\
         x=(0xf0f0f0f0&x)>>4+(0x0f0f0f0f&x),\
         x=(0xff00ff00&x)>>8+(0x00ff00ff&x),\
         x=x>>16+(0x0000ffff&x))
```

94. Compute the discrete log of an unsigned integer.

ANS.

```
#define discrete_log(h) \
 (h=(h>>1)|(h>>2),\
 h|=(h>>2),\
 h|=(h>>4),\
 h|=(h>>8),\
 h|=(h>>16),\
 h=(0xaaaaaaaa&h)>>1+(0x55555555&h),\
 h=(0xcccccccc&h)>>2+(0x33333333&h),\
 h=(0xf0f0f0f0&h)>>4+(0x0f0f0f0f&h),\
 h=(0xff00ff00&h)>>8+(0x00ff00ff&h),\
 h=(h>>16)+(0x0000ffff&h))
```

If I understand it right, log2(2) =1, log2(3)=1, log2(4)=2..... But this macro does not work out log2(0) which does not exist! How do you think it should be handled?

* 95. How do we test most simply if an unsigned integer is a power of two?

ANS. #define power_of_two(x) \ ((x)&&(~(x&(x-1))))

96. Set the highest significant bit of an unsigned integer to zero.

ANS. (from Denis Zabavchik) Set the highest significant bit of an unsigned integer to zero
```
#define zero_most_significant(h) \
(h&=(h>>1)|(h>>2),\
h|=(h>>2),\
h|=(h>>4),\
h|=(h>>8),\
h|=(h>>16))
```

97. Let f(k) = y where k is the y-th number in the increasing sequence of non-negative integers with the same number of ones in its binary representation as y, e.g. f(0) = 1, f(1) = 1, f(2) = 2, f(3) = 1, f(4) = 3, f(5) = 2, f(6) = 3 and so on. Given k >= 0, compute f(k).

Others

98. A character set has 1 and 2 byte characters. One byte characters have 0 as the first bit. You just keep accumulating the characters in a buffer. Suppose at some point the user types a backspace, how can you remove the character efficiently. (Note: You cant store the last character typed because the user can type in arbitrarily many backspaces)

99. What is the simples way to check if the sum of two unsigned integers has resulted in an overflow.

100. How do you represent an n-ary tree? Write a program to print the nodes of such a tree in breadth first order.

101. Write the 'tr' program of UNIX. Invoked as

tr -str1 -str2. It reads stdin and prints it out to stdout, replacing every occurance of str1[i] with str2[i].

e.g. tr -abc -xyz
to be and not to be <- input
to ye xnd not to ye <- output