C Programming Language

Introduction

1. What is the meaning of *#include<stdio.h>* and why this line is written in a C program? **[CU 2008 2011 2014]**
2. What is header file in C? Why it is needed? **[SU 2011]**
3. Distinguish between *syntax* and *semantics*. **[CU 2009 2010 2011]**
4. *A program is an algorithm coded in a high level language* - Justify. **[CU 2009]**

Variables, Data-types and Constants

1. What do you mean by tokens in C? What are the types of tokens in C?
2. Define keyword or identifier. How many keywords are there in ANSI C? State the rules for identifiers in C. **[SU 2011]**
3. Define constants. State the various types of constants supported by C.
4. Define an integer constant. What are the types of integer constants in C?
5. What is the utility of %u in printf()? **[SU 2011]**
6. Define a real constant. What are the various types of representing real constants in C?
7. What is the meaning of using %g in printf()? **[SU 2010]**.
8. What is the utility of "y.x" in printf()?
9. What do you mean by Single character constant and String constants?
10. Write a short note on – Backslash Character Constant.
11. Define a variable. State the rules for naming convention of a variable. What do you mean by variable declaration?
12. Distinguish between a variable and constant in C.
    Differentiate between variable declaration and variable definition in C. **[CU 2008 SU 2013]**
13. Define data types. What are the various data types supported by ANSI C? Why void is referred as a data types?
14. Explain the size and range of various data types available in C.
15. What do you mean by User-define type declaration?
16. What do you mean by Symbolic Constants? State the rules that are applied to a #define statement.
17. How you can declare a variable as constant? State the default values for constants. What happens when a variable is declared as volatile?

Operators and Expressions

1. Define an operator and an expression. Classify the operators in C.
2. Illustrate any four different types of operators in C. **[CU 2011]**
3. State the limitation of modulo operator in C. Does C have any exponentiation operator?
4. What do you mean by mixed mode arithmetic?
5. When and why do we get the error message *Lvalue required*? What do you mean by *Rvalue*? **[CU 2009 SU 2012]**
6. If x = 13 and y = -4 then what is the value of x % y? **[CU 2006]**

7. State the necessity of relational operators. What are the relational operators in C?

8. What are the logical operators in C? Define a logical expression. State the relative precedence of the relational and logical operators.

9. What do you mean by shorthand assignment operator? Explain with examples.

10. What do you mean by Increment and Decrement operators in C? State the rules for those operators. Explain the difference between i++ and ++i.

11. Consider the following to give the output -

```
main()
{
     int i = 5;

     printf("%d%d%d%d", i++, ++i, ++i, ++i);

}
```

[SU 2010]

12. Explain the utility of ternary operator.

13. Write a program to find the largest of three integers by ternary operator. **[CU 2016]**

14. What are the bitwise operators in C? Explain their utility with the help of a program.

15. What are the purposes of << and >> operators? In practise when they are utilised? **[SU 2014]**

16. Using bitwise operators write a program in C to count the number of 0's in the binary representation of an integer. **[CU 2009 2015]**

17. What will happen if the following code is executed on two integers -
$p = p \hat{q};$
$q = p \hat{q};$
$p = p \hat{q};$

18. Write a program in C that asks the user to enter an integer and checks if it is odd (using only bitwise operation). If the entered number is odd then output the number of 1's in its binary representation (using bitwise operation), otherwise out the integer is even. **[CU 2006]**

19. What are the special operators in C? Explain the use of sizeof() operators.

20. State the rules for evaluating an expression. Define type conversion.

21. Differentiate between implicit type conversion and explicit type conversion or type casting. Why type casting is necessary? **[CU 2010]**

Decision making

1. Define decision making. Differentiate between branching and looping.

2. What are the decision making and branching constructs or control statements available in C?

3. State the various types of *if* constructs available in C. Explain with the help of Flowcharts.

4. What do you mean by nesting of *if.....else* statements?

5. What do you mean by Dangling else problem?

6. What will be the output of the following?

```
void main()
{
      int i, j = 0;
      if ((i = ++j) == 1)
            printf("%d%d", i, j);
      j = 0;
      if ((i = j++) == 1)
      printf("%d%d", i, j);
}
```

[SU 2013]

7. What will be the output of the following code?

```
if (printf(""))
      printf("abc");
else
      if(printf("%d", printf("")))
            printf("xyz");
      else
      printf("www");
```

[SU 2013]

8. What do you mean by menu-driven programming? How it is achieved?

9. State the rules for switch statement.

10. which is more convenient to be used between *switch .. case* and *nested if .. else* **[SU 2013]**

11. What will be the output of the following program? Explain -

```
main()
{
      char ch = 'A';
      while(ch $<$= 'D' )
      {
            switch(ch)
            {
                  case 'A' : ch ++; continue;
                  case 'B' : ch ++; continue;
                  case 'C' : ch ++; continue;
                  case 'D' : ch ++; \\
            }
      }
      putchar(ch);
}
```

[CU 2011]

12. Distinguish between *break* and *continue* statements. **[CU 2009 2011 2014 2015]**

13. A *break* statement is used to exit a loop or a switch statement. What happens if the the *break* statement is used outside any such construct? **[CU 2009]**

14. Explain the use of ternary operator in decision making.
15. Define jump. Explain how forward and backward jumps are implemented in C.
16. Why do unconditional jumps have to be avoided in the programs? **[CU 2009]**
17. Define a program loop. What are the components of a loop? What are the different loop constructs available in C.?
18. Differentiate between entry controlled and exit controlled loops.
19. Classify loops based on the nature of control variable and the kind of value assigned to it. (Hint. – Counter controlled and Sentinel controlled loop)
20. Explain the basic syntax of while statement. Explain the basic syntax of do… while statement. Differentiate between both of them with example.
21. Explain the general syntax of for loop. Explain the execution flow of for loop.
22. Explain how for loop can be implemented by while loop? **[CU 2010]**
23. What is a null statement? Explain a typical use of it.
24. Compare for, while and do…while loops
25. Explain with example how more than one variable can be put at a time in for statement.
26. Explain jumps in loops. How break is used to jump out of a loop? How continue is used to skip a part of loop.
27. Explain the use of exit() to jump out of a program.
28. "Use of goto should be avoided". Explain a typical example where we find the application of goto becomes necessary.
29. How can we use for loop when the number of iterations are not in advance.
30. An integer is divisible by 11 if the sum of the digits in odd position is equal to the sum the digits in even position. Write a C program to find whether an integer is divisible by 11 using this rule. **[CU 2008 2014]**
31. Write a program in C that will find the GCD of two integers without using recursion. (*Hint: Use repeated subtraction*) **[CU 2014]**
32. Write a C program to find the LCM of two positive integers. One way to achieve this is to find the GCD of Integers and divide the product of the integers by this GCD. Write the algorithm to find the LCM directly, without the using GCD. **[CU 2006 2009]**
33. Write a program in C to compute the roots of a quadratic equation of the form $ax^2 + bx + c = 0$ given the values of a, b and c. **[CU 2007 2008]**
34. The Euler's number **e**, is used as the base of natural logarithm. It can be approximated using the formula

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + ... + \frac{1}{n!} \tag{1}$$

    WAP that approximates **e** using a loop that terminates when the difference between two successive values of **e** differ by less than 0.0000001. **[CU 2007]**
35. Use a do-while loop to compute the sum of every third integer beginning with 2 for all integer less than 100, that is , the summation of 2, 5, 8, 11, ..., etc. **[CU 2006]**

Array

1. Define an array. Why it is classified as a linear data structure?
2. Classify an array.
3. Why array indices are to be specified with the declared limits? What will happen if a reference outside the declared limits is specified?
4. Why does array index starts from 0 in C? **[SU 2010]**
5. *C does not do boundary checking on the elements of an array* - Justify. **[CU 2007 2008]**
6. Explain how an array reference is resolved in C? **[CU 2008]**
7. Are the expressions x[m+n] and m + n[x] equivalent? Justify. **[CU 2010]**
8. "Size of an array should be either a numeric constant or a symbolic constant." – Explain.
9. How one can initialize an array. Explain with examples of different dimensions.
10. Explain memory representation schemes of an array.
11. State the indexing formula for arrays of different dimensions.
12. Differentiate between Static and Dynamic allocation of an array.
13. State the disadvantages of an array.
14. What are subscripts? How they are written? What restrictions apply to the values that can be assigned to subscripts?
15. How does an array definition differ from that of an ordinary variable?
16. Can initial values be specified within an external array definition? Can they be specified within a static array definition? Can they be specified within an automatic array definition?
17. What value is automatically assigned to those array elements that are not explicitly initialized?
18. When are array declarations (in contrast to array definitions) required in a C program? How do such declarations differ from array definitions?
19. What is a Sparse Matrix? Classify sparse matrices
20. Write a program in C that reads an integer from standard input device and outputs the number of 1's in its binary representation to the standard output device. Note that the integer read is negative then its 2's complement will be considered as its binary response. **[CU 2008 2014]**
21. Write a C program to count the total no of 1's in the binary representation of an integer. **[CU 2011 SU 2009 2010]**
22. Write a program in C that asks the user to enter a list of integers. The programme has to output the largest value entered and the number of time it was entered. **[CU 2006]**
23. Write a program in C to check whether a five digit valid integer is a palindrome or not (for example: 15351 is a palindrome). Your program should include validation check for integers. **[CU 2011]**
24. Write a program in C to check whether the bit pattern of a given integer variables is palindrome or not. **[CU 2010]**

String

1. Differentiate between 'A' and "A". **[SU 2011 2012]**

2. Differentiate between
   char * str1 = "May God Bless You" and char str2[] = "May God Bless You". **[CU 2008 2014]**

3. Write a program to copy the content of the source string into the destination string, without using *strcpy* library function. **[CU 2010]**

4. WAP to eliminate multiple spaces in a string. For example, a string such as "The quiz was great ! ! !" should be converted to "The quiz was great !!!".

5. What will be the output of the following

```
main()
{
    char s[] = "Get organised! learn c!";
    printf("\n%s", &s[2]);
    printf("\n%s", s);
    printf("\n%s", &s);
    printf("\n%c", s[2]);
}
```

[SU 2013]

User Defined Functions

1. Define a function/ procedure/ sub-routine. State the flow control in a multi- function program with the help of a suitable diagram.

2. What do you mean by modular programming? State the characteristics of modular programming.

3. What do you mean by calling function and called function?

4. What are the components of function definition? What do you mean by function signature?

5. What is the default return type of a function?

6. By giving suitable examples explain function with built-in arguments, function with derived arguments and function with user-defined arguments.

7. "A function call is a postfix expression" – justify.

8. "If the actual parameters are more than the formal parameters, the extra parameters will be truncated" – justify.

9. "If the actual parameters are less than the formal ones then unmatched formal arguments will be initialized to some garbage" – justify.

10. What do you mean by function prototype? [**[CU 2016 SU 2010]**]

11. State the rules for function call.

12. Differentiate between a global prototype and a local prototype. What do you mean by scope of a function?

13. "Function prototyping is optional" – comment on this with suitable example.

14. Differentiate between formal and actual parameter of a function. **[SU 2011]**

15. *A function in C can not change the values of the arguments* - Do you agree or disagree with the statement? Give proper reasons. **[CU 2014]**

16. *Recursive functions are always better than their non-recursive counterparts* - Comment on the statement with proper justification. **[CU 2014]**

17. Categorize a user-defined function based on arguments passed to it and its return type.

18. Give an example of a function that returns multiple values.

19. Differentiate between *'pass by value'* and *'pass by reference'*. Which one is better? **[CU 2016 SU 2009 2012]**

20. Explain the rules for passing an array to a function.

21. If an array is passed to a function and some of its elements are modified by the function, are these alterations be identified in the parent program? Explain. **[CU 2013]**

22. What do you mean by storage class of a variable? Define scope, visibility and lifetime of a variable. **[CU 2008 2009 2011 2014 SU 2011 2012]**

23. Differentiate between automatic variable and global variable. What is the default value of a global variable?

24. What is a static variable? Give a example where main() is called within main(). **[CU 2015]**

25. What will happen if a global variable is defined after the definition of function where it is invoked?

26. What is the purpose of an *extern* variable? State the differences between an extern variable declaration and definition. **[CU 2010 2013]**

27. What is static variable?

28. Differentiate between *static variable* and *extern variable*. **[CU 2010]**

29. Differentiate between *auto* and *static* variables. **[SU 2011]**

30. What is register variable? **[CU 2011 SU 2012 2014]**

31. State the problems likely to be encountered when we pass global variables as parameters to a function.

32. Write a recursive function that returns the sum $1 + 2 + 3 + ... + n$, where n is a given positive integer. **[CU 2007]**

33. Write a program that will display an input line backwards by recursion. **[CU 2015]**

34. Consider the following C function

```
int f(int n)
{
      static int i = 1;
      if (n >= 5)
            return n;
      n = n + 1;
      i++;
      return f(n);
}
```

Give the value returned by calling f(1). **[CU 2014]**

35. Write a function to find the no of digits in an integer. **[SU 2011]**

36. Write a function that read an integer and return the string equivalent of it. **[CU 2014]**

37. Write a function in C that accepts a 'float' argument and returns the string equivalent of it. **[CU 2008 2011 SU 2011 2013]**

38. Convert the following function into a recursive function -

```
void function(int val)
{
int i;
for(i=1; i<val; i++ )
      printf("Value = \%d", i);
}
```

39. What will be the value returned by the following function when it is called with 11 as the value of the parameter?

```
int function (int n)
{
     if((n/2)!=0)
          return (fun(n/2) * 10 + n % 2);
     else
          return 1;
}
```

[CU 2006]

40. Write a function that, given a string, a width, and an empty string for output, centres the string in the output area. The function is to return 1 if the formatting is successful and 0 if any errors, such as string length greater than width, are found. **[CU 2007]**

41. Write a program to perform addition and multiplication of two matrices using separate functions and *switch* statement. **[SU 2010]**

42. Write a function that returns the number of days in between two dates, which are accepted by the function as its arrangements. **[CU 2007]**

43. Write a function that uses bitwise operators to print the binary representation of an integer. **[CU 2007]**

44. Write a recursive function that returns the greatest common divisor of its two arguments (positive integer) **[CU 2007]**

45. Consider the following

    (a) Four integer variable d, m, y and yd is defined within with in main ( )

    (b) Read the values of y (year) and yd (date of that year) inside main.

    (c) Write a function to compute date (d) and month (m) for these values of y and yd

    (d) Print the values of d and m from main function

    For example if y =2007 and yd=61 then d should be 2 and m should be 3; that is 2nd March, 2007 is the 61st day of the year (leap year should be considered). **[CU 2007]**

46. Write a program that reads an integer from the keyboard and then calls a recursive function to print it out in n reserve. For example, if the user enter 4762, it prints 2674. **[CU 2006 SU 2009]**

User Defined Data-types

1. Differentiate between a structure and an array.

2. Why a structure a called a heterogeneous data-type? Explain the concept of self referential structures in C with an example. **[CU 2009]**

3. What will happen if a structure member is initialized within the structure definition?

4. State the different procedures for defining a structure variable.

5. What is a type-defined structure? What is the utility of such a structure definition?

6. State the rules for initializing a structure variable at compile-time.

7. What are the different ways to access structure members?

8. What is nesting of structure?

9. What are the methods by which the values of a structure can be transferred from one function to another?

10. What is slack byte? What will be the output of the following struct student { int roll; char name[20], int name; }var; printf("size of the structure is %u", sizeof(struct student));

11. Differentiate between *structure* and *union* in C. **[CU 2008 2009 2010 2014]**

12. What are the utilities of *Union* in C? Explain with example. **[CU 2013]**

13. Differentiate between structure and union. **[CU 2015]**

14. Write a short note on Enumerated data type. **[CU 2016]**

15. Consider the following structure declaration.

```
struct nd
{
     int x;
     int y;
};

typedef struct nd complex;
```

Write three functions - *sum(), mult()* and *show()*, that will add two complex numbers, multiply two complex numbers and display a complex number. **[CU 2010 2015]**

16. In 2-dimensional Cartesian co-ordinate system a point has two components namely (abscissa, ordinate). A line is an object connecting two such points. Using C structures how do you represent

    - a point
    - a line

    Write a function in C that accept a line structure and returns

    - 1, if line is horizontal
    - 2, if the line is vertical
    - 3, if the line is oblique

    [CU 2007 SU 2009]

Pointers and Dynamic Memory Allocation

1. What is a pointer? **[CU 2010]**
2. Define – pointer variable, pointer value and pointer constant.
3. What is pointer definition? What do you mean by cast type of a pointer?
4. What is the utility of *address operator* and *indirection operator* or *dereferencing operator* ?
5. What do you mean by multiple indirections?
6. Discuss pointer arithmetic. **[CU 2010]**
7. What are the arithmetic operators that are permitted on pointers?
8. "Two pointers cannot be added" – justify.
9. "A value cannot be assigned to an arbitrary address i.e. &x = 10 is illegal" – justify.
10. *Adding 1 to a pointer increases the address stored in it by the size of the variable that it points (scale factor)* - Comment on this. **[CU 2007]**
11. What is the effect of *p++? Explain. **[CU 2011]**
12. The following code is given
    int a[20], *p;
    p = a;
    How can you print the 7th element through 'p'? **[CU 2006]**
13. What do you mean by the following declaration?
    int (p [10])(char a); **[CU 2015]**
14. How a pointer is related to an array 1D in C? **[CU 2008 SU 2011 2012]**
15. What is the output of the following code?

```
void main()
{
      int a[] = {2,4,6,8,10};
      int i, *ptr;
      for (ptr = a+4, i = o; i < 4; i++)
      printf("%d", ptr[-i]);
}
```

[SU 2013]

16. What will be the output of the following?

```
main()
{
      int a[] = {0, 1, 2, 3, 4};
      int *p [] = {a, a+1, a+2, a+3, a+4};
      printf("\n%u %u %d", p, *p, *(*p));
}
```

[SU 2014]

17. What do you mean by ragged array? **[CU 2015]**
18. What do you mean by function pointer? **[CU 2011]**
19. What is the purpose of pointer to functions? **[CU 2010]**
20. Give an example of a function that returns a pointer.
21. What do you mean by incompatibility of pointers?
22. What is void pointer? Why it is needed? **[CU 2010 SU 2012]**
23. Is there any difference between a void pointer and a NULL pointer? **[CU 2015]**
24. Given below are two two different definitions of the function search()

```
void search (int *m[], int x){ }
void search (int ** m, int x){ }
```

    Are they equivalent? Justify.

25. Define a structure called *student* with fields name, dob, marks. Dynamically allocate a variable of type *student*. **[SU 2009]**
26. In the following program, how can you print 11 using 'ptr'?

```
void main()
{
      int arr[] = { -2, 5, 0, 29, 11, 7, 10};
      char *ptr;
      ptr = (char *) arr;
}
```

[CU 2006]

27. Write a program in C that contains an array pointers to some strings. The program is supposed to search out the string with maximum length. **[CU 2013]**
28. Suppose a pointer is holding the starting address of an array. Write a program and pass the pointer to a function to increment the pointer so that any arbitrary element within the array can be accessed according to user's choice. **[CU 2013]**
29. *Whatever we can do using an array can also be done by pointers in C* - Justify the statement with suitable examples. What benefits result if we do so? **[CU 2007 SU 2013]**
30. What do you mean by dynamic memory allocation? Explain the storage of a C program in memory by drawing a suitable diagram.

31. Explain the syntax of the functions – malloc(), calloc() and realloc(). **[CU 2010]**
32. Differentiate between malloc() and calloc(). **[CU 2010 2016 SU 2011]**
33. Differentiate between memory releasing strategies by free() and release by storage class.
34. What do you mean by memory leak in C? How it can be avoided? **[CU 2016]**

File Handling

1. Why file handling programs are useful in C?
2. What is *file pointer*? **[CU 2010 2015]**
3. What is the significance of EOF?
4. When a program is terminated, all the files used by the program are automatically closed. Why is it then necessary to close a file during the execution of the program?
5. Distinguish between the following functions -
   (a) getc() and getchar()
   (b) putchar() and putch() **[CU 2016]**
   (c) fgetc() and fgets()
   (d) printf() and fprintf()
   (e) feof() and ferror()
   (f) fwrite() and fprintf()
   (g) fscanf() and fread()
6. How does it differ opening a file in write mode than opening it in append mode? **[SU 2009]**
7. State some of the typical error situations that may occur during I/O operations on a file.
8. Write short notes on – rewind() and ftell().
9. By stating the general format for fseek() function explain its utility.
10. Differentiate between rewind(fp) & fseek(fp,0L,0). **[CU 2006]**
11. What will be the output of the following

```
    for(i=0;i<=5;i++)
    {
         fscanf(stdin, "%s", name );
         fprintf(fp, "%s", name);
    }
```

12. WAP to demonstrate the use of feof() and ferror().
13. Can we read from a file and write to the same file without resetting the file pointer?
14. Explain the signature of the main() for programs with command line arguments.
15. Write a C program that will read a file containing integers. Find the odd and even numbers from the file and keep those numbers into two new files “Odd.txt” and “Even.txt”. **[SU 2011]**
16. Write a program that will merge the contents of two files into another. **[SU 2013]**
17. Write a program in C that will take the name of a text file from command line and will print all the word string with a vowel into another file name vowel.txt. **[CU 2010 2016]**
18. Write a program in C to print the name of your program i.e. the name of the program that is executing. **[CU 2009]**
19. If the standard input is **part** , then what will be the output of the following program?

```
#include<stdio.h>
main()
{
      mystery();
}

mystery()
{
      int c;
      if((c = getchar())! = EOF)
      {
            mystery();
            putchar(c);
      }
}
```

20. Write a program in C that will copy the content of one file, say *file1* into another, say *file2*. **[SU 2009]**

21. Consider the following -

```
#include<stdio.h>
int main(int argc, char * argv[])
{
            while(--argc)
                  printf("%s%c", * ++ argv, argc > ? "":"\n");
            return 0;
}
```

Suppose the above program's name is echo and you have given good morning as argument. What will be the output? Assume the program is modified as -

```
#include<stdio.h>
int main()
{
            int argc = 3;
            char * argv[] = {"echo", "good", "morning"};
            while(--argc)
                  printf("%s%c", * ++ argv, argc > ? "":"\n");
            return 0;
}
```

Will the output change? **[CU 2015]**

Preprocessor

1. What is preprocessor? **[CU 2013 SU 2010]**
2. What is the purpose of a *macro*? Explain with suitable examples. **[CU 2010]**
3. Do you think macros are sometimes advantageous over functions in C? Explain with suitable example. **[CU 2013]**
4. Differentiate between a macro and a function. **[CU 2011 2015]**
5. Give an example when macro should not be used in a C program. **[SU 2014]**
6. What do you mean by a macro call?
7. Differentiate between a macro and a macro-within-a-macro.
8. "Macro inside a string does not get replaced" – justify with suitable example.
9. "A macro definition can include more than a simple constant value" – justify with a example.
10. "Use of parenthesis is recommended in defining a macro" – Explain the justification.

11. Differentiate between #include<filename> and #include “filename”. **SU 2010**

12. What do you mean by compiler control directive?

13. Differentiate between #elif and #if

14. Can a C program be included in a C program by file inclusion directive?

15. What is the output the following code -

```
#define min(x,y) ((x)<=(y)):(x) : (y)
#define max(x,y) ((x)>=(y)) : (x) :(y)

int a = 14, b = 21, c = 54, d = 45;
printf("Large = %d Small = %d", max(c,d) - 1, min(a,b) + 1);
```

[CU 2008 2009]

16. Give the output of the following program segment -
#define PRINT(x,y,z) printf(“x = %d y = %d z = %d”, x,y,z)
main()
{
int x, y, z;
x = y = z = 1;
$++x||++y\&\&++z$;
PRINT(x,y,z);
x = y = z = 1;
$++x\&\&++y||++z$;
PRINT(x,y,z);
x = y = z = -1;
$++x\&\&++y||++z$;
PRINT(x,y,z);
x = y = z -1;
$++x||++y\&\&++z$;
PRINT(x,y,z);
}

17. Give the output of the following program or specify the error, if any

```
#define multiply(a,b) a*b
main()
{
      int a = 4, b = 2;
      printf("%d", multiply(a+b, a-b));
}
```

18. Suppose that x, y and z are variables of type **float** in a program. If these variables have the values 1.1, 2.2 and 3.3 respectively, then the statement **PRN3(x,y,z)**; should cause the line ***x has value 1.1, y has value 2.2 and z has value 3.3*** to be printed. Write the macro definition for ***PRN3()***. **[CU 2006]**

19. Consider the following -

```
#include<stdio.h>
#define SQUARE(x) x * x

int main()
{
      int a, b, c;

      a = 2;
      b = 3;
      c = SQUARE(a + b);
      printf("c = %d\n", c);

      return 0;
}
```

Identify the problem, if there is any, in the above code and suggest a solution. **[CU 2016]**