What is C?

C is a general-purpose computer programming language developed between 1969 and 1973 by Dennis Ritchie at the Bell Telephone Laboratories for use with the Unix operating system

Although C was designed for implementing system software, it is also widely used for developing portable application software.

C is one of the most widely used programming languages of all time and there are very few computer architectures for which a C compiler does not exist. C has greatly influenced many other popular programming languages, most notably C++, which began as an extension to C.

Difference between C & C++?

1. C follows the procedural programming paradigm while C++ is a multi-paradigm language(procedural as well as object oriented).

In case of C, importance is given to the steps or procedure of the program while C++ focuses on the data rather than the process.
Also, it is easier to implement/edit the code in case of C++ for the same reason.

2. In case of C, the data is not secured while the data is secured(hidden) in C++

This difference is due to specific OOP features like Data Hiding which are not present in C.

3. C is a low-level language while C++ is a middle-level language (Relatively, Please see the discussion at the end of the post)

C is regarded as a low-level language(difficult interpretation & less user friendly) while C++ has features of both low-level(concentration on whats going on in the machine hardware) & high-level languages(concentration on the program itself) & hence is regarded as a middle-level language.

4. C uses the top-down approach while C++ uses the bottom-up approach

In case of C, the program is formulated step by step, each step is processed into detail while in C++, the base elements are first formulated which then are linked together to give rise to larger systems

5. C is function-driven while C++ is object-driven

Functions are the building blocks of a C program while objects are building blocks of a C++ program

6. C++ supports function overloading while C does not

Overloading means two functions having the same name in the same program. This can be done only in C++ with the help of Polymorphism(an OOP feature)

7. We can use functions inside structures in C++ but not in C.

In case of C++, functions can be used inside a structure while structures cannot contain functions in C.

8. The NAMESPACE feature in C++ is absent in case of C

C++ uses NAMESPACE which avoid name collisions. For instance, two students enrolled in the same university cannot have the same roll number while two students in different universities might have the same roll number. The universities are two different namespace & hence contain the same roll number(identifier) but the same university(one namespace) cannot have two students with the same roll number(identifier)

9. The standard input & output functions differ in the two languages

C uses scanf & printf while C++ uses cin>> & cout<< as their respective input & output functions

10. C++ allows the use of reference variables while C does not

Reference variables allow two variable names to point to the same memory location. We cannot use these variables in C programming.

11. C++ supports Exception Handling while C does not.

C does not support it "formally" but it can always be implemented by other methods. Though you don't have the framework to throw & catch exceptions as in C++.

In printf() Function- What is the difference between "printf(...)" and "sprintf(...)"?

sprintf(...) writes data to the character array whereas printf(...) writes data to the standard output device.

Compilation How to reduce a final size of executable?

Size of the final executable can be reduced using dynamic linking for libraries

---

Write out a function that prints out all the permutations of a string. For example, abc would give you abc, acb, bac, bca, cab, cba.

```
void PrintPermu (char *sBegin, char* sRest) {
     int iLoop;
     char cTmp;
     char cFLetter[1];
     char *sNewBegin;
     char *sCur;
     int iLen;
     static int iCount;
     iLen = strlen(sRest);

     if (iLen == 2) {
          iCount++;
          printf("%d: %s%s\n",iCount,sBegin,sRest);
          iCount++;
          printf("%d: %s%c%c\n",iCount,sBegin,sRest[1],sRest[0]);
          return;
     } else if (iLen == 1) {
          iCount++;
          printf("%d: %s%s\n", iCount, sBegin, sRest);
          return;
     } else {
          // swap the first character of sRest with each of
          // the remaining chars recursively call debug print
          sCur = (char*)malloc(iLen);
          sNewBegin = (char*)malloc(iLen);
          for (iLoop = 0; iLoop <>
          strcpy(sCur, sRest);
          strcpy(sNewBegin, sBegin);
```

```
          cTmp = sCur[iLoop];
          sCur[iLoop] = sCur[0];
          sCur[0] = cTmp;
          sprintf(cFLetter, "%c", sCur[0]);
          strcat(sNewBegin, cFLetter);
          debugprint(sNewBegin, sCur+1);
     }
}

void main() {
     char s[255];
     char sIn[255];
     printf("\nEnter a string:");
     scanf("%s%*c",sIn);
     memset(s,0,255);
     PrintPermu(s, sIn);
}
```

What is assignment operator?

Default assignment operator handles assigning one object to another of the same class. Member to member copy (shallow copy)

What is conversion operator?

Class can have a public method for specific data type conversions.

For example:

```
class Boo
{
  double value;
  public:
    Boo(int i )
    operator double()
    {
     return value;
    }
};

Boo BooObject;

double i  = BooObject; // assigning object to variable i of type double. now conversion  operator gets called to assign the value.
```

What is diff between malloc()/free() and new/delete?

malloc allocates memory for object in heap but doesn't invoke object's constructor to initiallize the object.

new allocates memory and also invokes constructor to initialize the object.

malloc() and free() do not support object semantics

Does not construct and destruct objects

```
string * ptr = (string *)(malloc (sizeof(string)))
```

Are not safe

Does not calculate the size of the objects that it construct

Returns a pointer to void

```
int *p = (int *) (malloc(sizeof(int)));

int *p = new int;
```

Are not extensible

new and delete can be overloaded in a class

"delete" first calls the object's termination routine (i.e. its destructor) and then releases the space the object occupied on the heap memory. If an array of objects was created using new, then delete must be told that it is dealing with an array by preceding the name with an empty []:-

```
Int_t *my_ints = new Int_t[10];

...

delete []my_ints;
```

---

what is the diff between "new" and "operator new" ?

"operator new" works like malloc.

What is difference between template and macro??

There is no way for the compiler to verify that the macro parameters are of compatible types. The macro is expanded without any special type checking.

If macro parameter has a postincremented variable (like c++), the increment is performed two times.

Because macros are expanded by the preprocessor, compiler error messages will refer to the expanded macro, rather than the macro definition itself. Also, the macro will show up in expanded form during debugging.

For example

Macro:

```
#define min(i, j) (i < j ? i : j)


template:


template<class T>

T min (T i, T j)
```

```
{
     return i < j ? i : j;
}
```

What is static variable?

Static variable is a variable that has visibility of a local variable and life time of an external variable. It is stored in main memory and has default value zero.

Example:

```
#include <stdio.h>

//program in file f1.c

void count(void) {
     static int count1 = 0;
     int count2 = 0;
     count1++;
     count2++;
     printf("\nValue of count1 is %d, Value of count2 is %d", count1, count2);
}

/*Main function*/

int main(){
    count();
    count();
    count();
    return 0;
}

Output:
Value of count1 is 1, Value of count2 is 1
Value of count1 is 2, Value of count2 is 1
Value of count1 is 3, Value of count2 is 1
```

Where are the auto variables stored?

Auto variables are stored in main memory and their default value is a garbage value.

Where does global, static, local, register variables and C Program instructions get stored?

Global, static, local : In main memory

Register variable: In registers

C program : In main memory

Why Preincrement operator is faster than Postincrement?

Evaluation of any expression is from left to right. Preincrement is faster because it doesn't need to save the current value for next instruction whereas Postincrement needs to saves current value to be incremented after execution of current instruction.

In header files whether functions are declared or defined?

Functions in header file are declared.

Difference between arrays and linked list?

Major differences between arrays and linked lists are:

1. In array consecutive elements are stored in consecutive memory locations whereas in linked list it not so.
2. In array address of next element is consecutive and whereas in linked list it is specified in the address part of each node.
3. Linked List makes better use of memory than arrays.
4. Insertion or deletion of an element in array is difficult than insertion or deletion in linked list.

What is the use of typedef?

Major uses of typedef are:

1. It increases the portability.
2. It simplify the complex declaration and improve readability of the program.

What are the differences between malloc() and calloc()?

A malloc( ) function allocates a block of memory of the specified size and returns a pointer of specified data type whereas a calloc( ) function allocates a space for an array of elements , initializes them to zero and then returns a pointer to the memory.

What do the ‘c’ and ‘v’ in argc and argv stand for?

c stands for counter and v stands for vector.

Is function declarations are used only for compilation and not get stored in the .EXE File ?

Yes

Are the variables argc and argv are local to main?

Yes

What is a pointer?

A pointer is a special variable in C language meant just to store address of any other variable or function. Pointer variables unlike ordinary variables cannot be operated with all the arithmetic operations such as '*','%' operators.
It follows a special arithmetic called as pointer arithmetic.

A pointer is declared as:

```
int *ap;

int a = 5;
```

In the above two statements an integer a was declared and initialized to 5. A pointer to an integer with name ap was declared.

Next before ap is used

```
ap=&a;
```

This operation would initialize the declared pointer to int. The pointer ap is now said to point to a.

Operations on a pointer:

Dereferencing operator ' * ':

This operator gives the value at the address pointed by the pointer . For example after the above C statements if we give

```
printf("%d",*ap);
```

Actual value of a that is 5 would be printed. That is because ap points to a.

Addition operator ' + ':

Pointer arithmetic is different from ordinary arithmetic.

```
ap=ap+1;
```

Above expression would not increment the value of ap by one, but would increment it by the number of bytes of the data type it is pointing to. Here ap is pointing to an integer variable hence ap is incremented by 2 or 4 bytes depending upon the compiler.

A pointer is a special variable in C language meant just to store address of any other variable or function. Pointer variables unlike ordinary variables cannot be operated with all the arithmetic operations such as '*','%' operators. It follows a special arithmetic called as pointer arithmetic.

A pointer is declared as:

```
int *ap;

int a = 5;
```

In the above two statements an integer a was declared and initialized to 5. A pointer to an integer with name ap was declared.

Next before ap is used

```
ap=&a;
```

This operation would initialize the declared pointer to int. The pointer ap is now said to point to a.

What is a structure?

A structure is a collection of pre-defined data types to create a user-defined data type. Let us say we need to create records of students. Each student has three fields:

```
int roll_number;
char name[30];
int total_marks;
```

This concept would be particularly useful in grouping data types. You could declare a structure student as:

```
struct student {
    int roll_number;
    char name[30];
    int total_marks;
} student1, student2;
```

The above snippet of code would declare a structure by name student and it initializes two objects student1, student2. Now these objects and their fields could be accessed by saying student1.

roll_number for accesing roll number field of student1 object, similarly student2.

name for accesing name field of student2 object.

What is recursion? Write a program using recursion (factorial)?

A function is called 'recursive' if a statement within the body of a function calls the same function. It is also called 'circular definition'. Recursion is thus a process of defining something in terms of itself.

Example

To calculate the factorial value using recursion.

```
#include <stdio.h>
int fact(int n);

int main() {
 int x, i;
 printf("Enter a value for x: \n");
 scanf("%d", &x);
 i = fact(x);
 printf("\nFactorial of %d is %d", x, i);
 return 0;
}

int fact(int n) {
 /* n=0 indicates a terminating condition */
 if (n <= 0) {
```

```
  return (1);
 } else {
  /* function calling itself */
  return (n * fact(n - 1));
  /*n*fact(n-1) is a recursive expression */
 }
}
```

Output:

Enter a value for x:

4

Factorial of 4 is 24

Explanation:

fact(n) = n * fact(n-1)

If n=4

fact(4) = 4 * fact(3) there is a call to fact(3)

fact(3) = 3 * fact(2)

fact(2) = 2 * fact(1)

fact(1) = 1 * fact(0)

fact(0) = 1

fact(1) = 1 * 1 = 1

fact(2) = 2 * 1 = 2

fact(3) = 3 * 2 = 6

Thus fact(4) = 4 * 6 = 24

Terminating condition(n <= 0 here;) is a must for a recursive program. Otherwise the program enters into an infinite loop.

To which numbering system, can the binary number 1101100100111100 be easily converted to?

1101100100111100 can be easily converted to hexadecimal numbering system. Hexa-decimal integer constants consist of combination of digits from 0 to 9 and alphabets 'A' to 'F'. The alphabets represent numbers 10 to 15 respectively. Hexa-decimal numbers are preceeded by '0x'.

```
1101,1001,0011,1100
      1101 = D
      1001 = 9
      0011 = 3
      1100 = C
1101,1001,0011,1100 = 0xD93C
```

Thus the given binary number 1101100100111100 in hexadecimal form is 0xD93C

What are the differences between structures and unions?

Structures and Unions are used to store members of different data types.

a) **Declaration:**

```
struct {    data type member1;    data type member2;   };
union {    data type member1;    data type member2;   };
```

b) Every structure member is allocated memory when a structure variable is defined. The memory equivalent to the largest item is allocated commonly for all members in union.

Example:

```
struct emp {  char name[5];  int age;  float sal; }; struct emp e1;
```

Memory allocated for structure is 5+4+4=13 bytes(assuming sizeof int is 4, float is 4, char is 1). 5 byte for name, 4 bytes for age and 4 bytes for sal.

```
union emp1 {  char name[5];  int age;  float sal; }; union emp1 e2;
```

Memory allocated to a union is equal to size of the largest member. In this case, char name[5] is the largest-sized member. Hence memory allocated to this union is 5 bytes.

c) All structure variables can be initialized at a time

```
struct st {  int a;  float b; }; struct st s = { .a=4, .b=10.5 };
```

Structure is used when all members are to be independently used in a program.

Only one union member can be initialized at a time

```
union un {  int a;  float b; }; union un un1 = { .a=10 };
```

Union is used when members of it are not required to be accessed at the same time

---

What are the advantages of using unions?

Union is a collection of data items of different data types.
It can hold data of only one member at a time though it has members of different data types.

If a union has two members of different data types, they are allocated the same memory. The memory allocated is equal to maximum size of the members. The data is interpreted in bytes depending on which member is being accessed.

Example:

```
union pen {  char name;  float point; };
```

Here name and point are union members. Out of these two variables, 'point' is larger variable which is of float data type and it would need 4 bytes of memory. Therefore 4 bytes space is allocated for both the variables. Both the variables have the same memory location. They are accessed according to their type.
Union is efficient when members of it are not required to be accessed at the same time.

What are the advantages of using pointers in a program?

Major advantages of pointers are:

1. It allows management of structures which are allocated memory dynamically.
2. It allows passing of arrays and strings to functions more efficiently.

3. It makes possible to pass address of structure instead of entire structure to the functions.
4. It makes possible to return more than one value from the function.

What is encapsulation?

Containing and hiding information about an object, such as internal data structures and code. Encapsulation isolates the internal complexity of an object's operation from the rest of the application. For example, a client component asking for net revenue from a business object need not know the data's origin.

What is inheritance?

Inheritance allows one class to reuse the state and behavior of another class. The derived class inherits the properties and method implementations of the base class and extends it by overriding methods and adding additional properties and methods.

What is Polymorphism?

Polymorphism allows a client to treat different objects in the same way even if they were created from different classes and exhibit different behaviors.

You can use implementation inheritance to achieve polymorphism in languages such as C++ and Java.

Base class object's pointer can invoke methods in derived class objects.

You can also achieve polymorphism in C++ by function overloading and operator overloading.

What is constructor or ctor?

Constructor creates an object and initializes it. It also creates vtable for virtual functions. It is different from other methods in a class.

What is destructor?

Destructor usually deletes any extra resources allocated by the object.

What is default constructor?

Constructor with no arguments or all the arguments has default values.

What is copy constructor?

Constructor which initializes the it's object member variables (by shallow copying) with another object of the same class. If you don't implement one in your class then compiler implements one for you.

For example:

```
Boo Obj1(10); // calling Boo constructor

Boo Obj2(Obj1); // calling boo copy constructor

Boo Obj2 = Obj1;// calling boo copy constructor
```

When are copy constructors called?

Copy constructors are called in following cases:

1. when a function returns an object of that class by value
2. when the object of that class is passed by value as an argument to a function
3. when you construct an object based on another object of the same class
4. When compiler generates a temporary object

What are all the implicit member functions of the class? Or what are all the functionswhich compiler implements for us if we don't define one.?

```
default ctor

copy ctor

assignment operator

default destructor

address operator
```

What is conversion constructor?

constructor with a single argument makes that constructor as conversion ctor and it can be used for type conversion.

For example:

```
class Boo  {

  public:

    Boo( int i );

};


Boo BooObject = 10 ; // assigning int 10 Boo object
```

What will print out?

```
main()
{
    char *p1="name";
    char *p2;
```

```
    p2=(char*)malloc(20);
    memset (p2, 0, 20);
    while(*p2++ = *p1++);
    printf("%sn",p2);
}
```

Empty string

What will be printed as the result of the operation below:

```
main()
{
  int x=20,y=35;
  x=y++ + x++;
  y= ++y + ++x;
  printf("%d%dn",x,y);
}
```

5794

What will be printed as the result of the operation below:

```
main()
{
  int x=5;
  printf("%d,%d,%dn",x,x< <2,x>>2);
}
```

5,20,1

What will be printed as the result of the operation below:

```
#define swap(a,b) a=a+b;b=a-b;a=a-b;

void main()
{
  int x=5, y=10;
  swap (x,y);
  printf("%d %dn",x,y);
  swap2(x,y);
  printf("%d %dn",x,y);
}

int swap2(int a, int b)
{
```

```
    int temp;
    temp=a;
    b=a;
    a=temp;
    return 0;
}
```

10, 5

10, 5

What will be printed as the result of the operation below:

```
main()
{
    char *ptr = " Cisco Systems";
    *ptr++; printf("%sn",ptr);
    ptr++;
    printf("%sn",ptr);
}
```

Cisco Systems

isco systems

What will be printed as the result of the operation below:

```
main()
{
    char s1[]="Cisco";
    char s2[]= "systems";
    printf("%s",s1);
}
```

Cisco

What will be printed as the result of the operation below:

```
main()
{
    char *p1;
    char *p2;

    p1=(char *)malloc(25);
    p2=(char *)malloc(25);

    strcpy(p1,"Cisco");
    strcpy(p2,"systems");
    strcat(p1,p2);
```

```
   printf(“%s”,p1);
}
```

Ciscosystems

The following variable is available in file1.c, who can access it?

static int average;

all the functions in the file1.c can access the variable.

What will be the result of the following code?

```
#define TRUE 0 // some code

while(TRUE)
{
    // some code
}
```

This will not go into the loop as TRUE is defined as 0.

What will be printed as the result of the operation below:

```
int x;
int modifyvalue()
{
   return(x+=10);
}
int changevalue(int x)
{
   return(x+=1);
}
void main()
{
   int x=10;
   x++;
   changevalue(x);
   x++;
   modifyvalue();
   printf("First output:%dn",x);
   x++;
   changevalue(x);
   printf("Second output:%dn",x);
```

```
   modifyvalue();
   printf("Third output:%dn",x);
}
```

12 , 13 , 13

What will be printed as the result of the operation below:

```
main()
{
   int x=10, y=15;
   x = x++;
   y = ++y;
   printf("%d %dn",x,y);
}
```

11, 16

What will be printed as the result of the operation below:

```
main()
{
   int a=0;
   if(a==0)
      printf("Cisco Systemsn");
      printf("Cisco Systemsn");
}
```

Two lines with "Cisco Systems" will be printed.

How to print below pattern?

1

2 3

4 5 6

7 8 9 10

Program:

```
#include <stdio.h>


int main() {
 int i, j, ctr = 1;
 for (i = 1; i < 5; i++) {
```

```
    for (j = 1; j <= i; j++){
      printf("%2d ", ctr++);
    }
    printf("\n");
 }
 return 0;
}
```

Explanation:

There are two loops, a loop inside another one. Outer loop iterates 5 times. Inner loop iterates as many times as current value of i. So for first time outer loop is executed, inner loop is executed once. Second time the outer loop is entered, inner loop is executed twice and so on. And every time the program enters inner loop, value of variable ctr is printed and is incremented by 1. %2d ensures that the number is printed in two spaces for proper alignment

---

How to swap two numbers using bitwise operators?

Program:

```
#include <stdio.h>
int main() {
 int i = 65;
 int k = 120;
 printf("\n value of i=%d k=%d before swapping", i, k);
 i = i ^ k;
 k = i ^ k;
 i = i ^ k;
 printf("\n value of i=%d k=%d after swapping", i, k);
 return 0;
}
```

Explanation:

i = 65; binary equivalent of 65 is 0100 0001

k = 120; binary equivalent of 120 is 0111 1000

i = i^k;

i...0100 0001

k...0111 1000

---

val of i = 0011 1001

---

k = i^k

i...0011 1001

k...0111 1000

---

val of k = 0100 0001 binary equivalent of this is 65

---------(that is the initial value of i)

i = i^k

i...0011 1001

k...0100 0001

---

val of i = 0111 1000 binary equivalent of this is 120

---------(that is the initial value of k)

Do you memory representation of float data type in c?

Float numbers are stored in exponential form i.e.

```
(Mantissa)*10^ (Exponent)
```

Here * indicates multiplication and ^ indicates power.

In memory only Mantissa and Exponent is stored not *, 10 and ^.

Total size of float data type: 32 bit

Those bits are used in following manner:

```
Exponent bit: 8

Mantissa bit: 24

Mantissa is signed number, so 24 bit are used as:

Mantissa_sign bit: 1

Mantisaa_data bit: 23
```

For only mantissa:

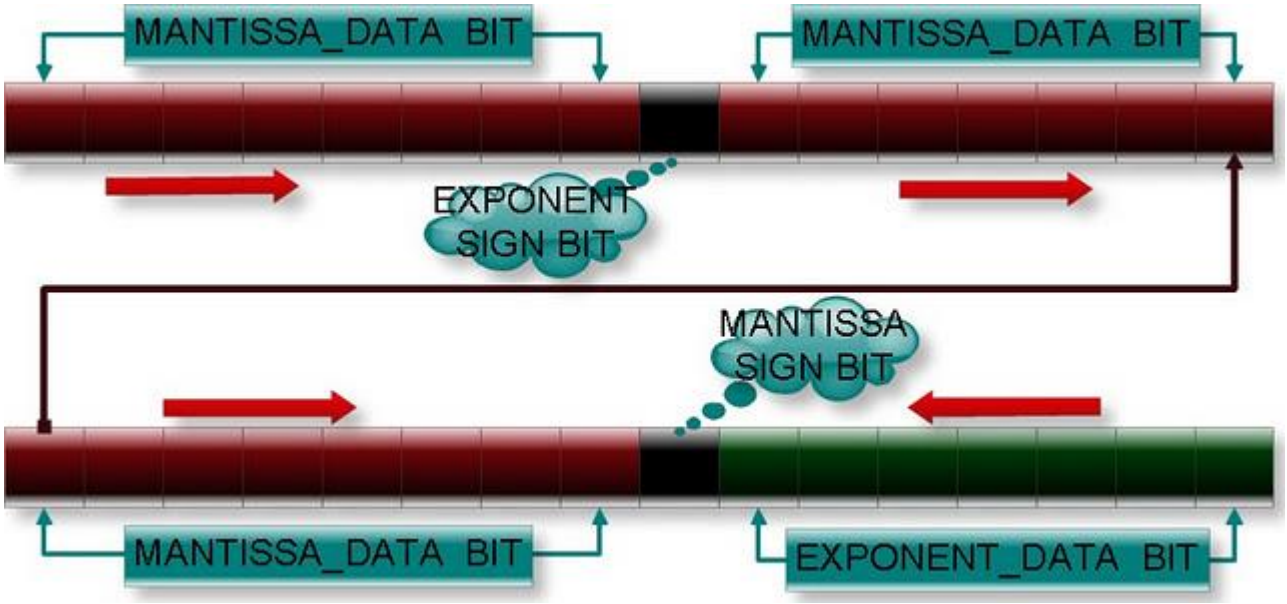
Mantissa_sign bit will zero if number is positive and Mantissa_sign bit will one if number is negative.

Exponent is also signed number, So 8 bit are used as:

```
Exponent_sign bit: 1

Exponent_data bit: 7
```

Following figure illustrate how floating point number is stored in memory.



Five important rules:

Rule 1: To find the mantissa and exponent, we convert data into scientific form.

Rule 2: Before the storing of exponent, 127 is added to exponent.

Rule 3: Exponent is stored in memory in first byte from right to left side.

Rule 4: If exponent will negative number it will be stored in 2's complement form.

Rule 5: Mantissa is stored in the memory in second byte onward from right to left side. Example:

Memory representation of:

```
float a = -10.3f;
```

For this you have to follow following steps:

step1: convert the number (10.3) into binary form

Binary value of 10.3 is: 1010.0100110011001100110011001100110011...

step2: convert the above binary number in the scientific form. Scientific form of 1010.0100110011001100110011001100110011...= 1.0100100110011001100110011001100 11001100110011...*10^3

Note: First digit i.e. 1, decimal point symbol, base of power i.e. 10, power symbol ^ and multiplication symbol * are not stored in the memory.

Step3: find exponent and mantissa and signed bit

Mantissa_data bit in binary = 0100100 11001100 11001101 (Only first 23 bit from left side)

Mantissa_sign bit: 1 (Since it is a negative number)

Exponent in decimal: 3

Question:

Why we have taken right most bit of mantissa_data bit one instead of zero?

Step5: Add 127 in the exponent and convert in the binary number form.

(Why 127? since size of exponent_data bit is 7 and maximum possible number in seven bit will 1111111 in binary or 127 in decimal)

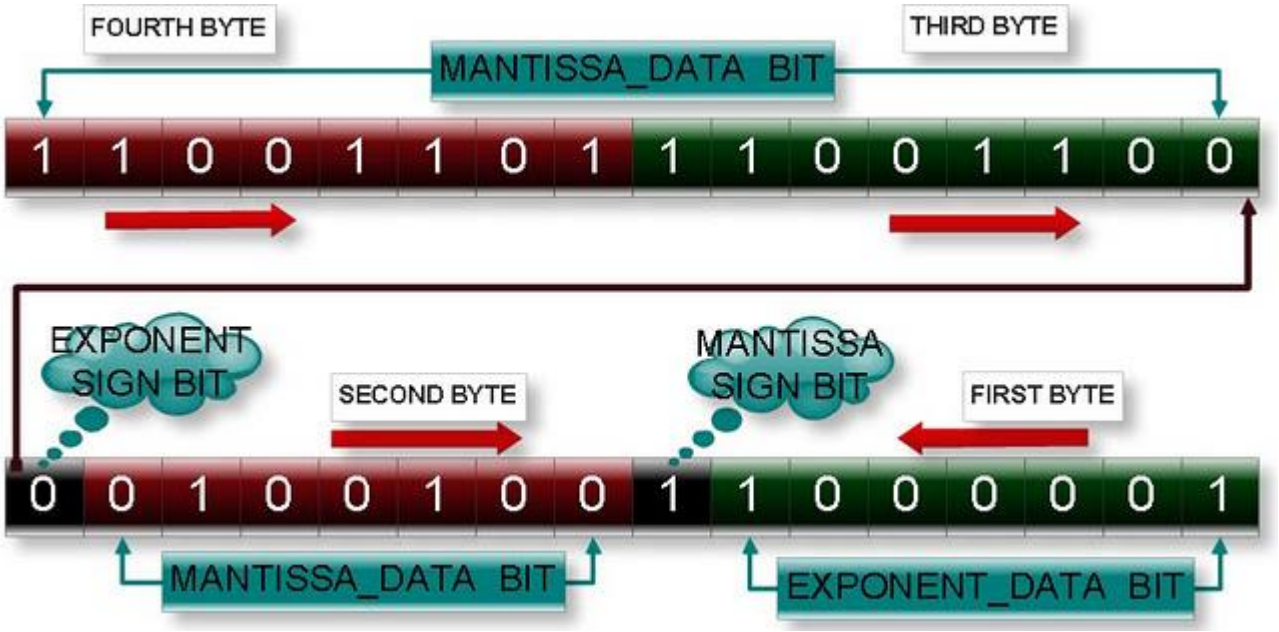
Exponent= 127+3=130

Binary value of 130 in eight bit: 1000001 0

Exponent_data bit: 1000001 (Take first seven bit from left side)

Exponent_sign bit: 0 (Take rightmost bit)

Step6: Now store the Mantissa_data bit, Mantissa_sign bit, Exponent_data bit and Exponent_sign bit at appropriate location as shown in the following figure.



Note: Mantissa_data bits are stored from left to right while Exponent_data bits are stored from right to left.

How to check above memory representation is correct?

Answer:

We will take one char pointer and visit each byte of a float number and observe the output.

```
#include<stdio.h>
int main(){
int i;
float f=-10.3f;
char *p=(char *)&f;
for(i=0;i<4;i++)
printf("%d ",*p++);
return 0;
}

Output: -51 -52 36 -63
Explanation:
Binary value of -51 in eight bit: 11001101
Binary value of -52 in eight bit: 11001100
Binary value of 36 in eight bit: 00100100
Binary value of -63 in eight bit: 11000001
```

This is exactly same as which we have represented in memory in the above figure.

Write a c program which changes the position of cursor?

```
#include<dos.h>
#include<stdio.h>
```

```
      int main(){
      union REGS i,o;
      i.h.ah=2; //positioning the cursor
      i.h.bh=0;
      i.h.dh=30;
      i.h.dl=45;
      int86(0x10,&i,&o);
      printf("World");
      return 0;
}
```

Do you know pragma directives in c?

Pragma is implementation specific directive i.e each pragma directive has different implementation rule and use. There are many type of pragma directive and varies from one compiler to another compiler .If compiler does not recognize particular pragma the it simply ignore that pragma statement without showing any error or warning message and execute the whole program assuming this pragma statement is not present.
For example suppose there is any pragma directive is #pragma world.

```
#include<stdio.h>
#pragma world
int main(){
    printf("C is powerful language ");
    return 0;
}
```

Output: C is powerful language
Explanation:
Since #pragma world is unknown for Turbo c 3.0 compilers so it will ignore this directive without showing any error or warning message and execute the whole program assuming #pragma world statement is not present.

List of pragma directives in turbo c 3.0:

1. #pragma startup
2. #pragma exit
3. #pragma warn
4. #pragma option
5. #pragma inline
6. #pragma argsused
7. #pragma hdrfile
8. #pragma hdrstop
9. #pragma saveregs

What is array of pointers in c?

Array whose content is address of another variable is known as array pointers. For example:

```
#include<stdio.h>
int main(){
```

```
float a=0.0f,b=1.0f,c=2.0f;
    float * arr[]={&a,&b,&c};
    b=a+c;
    printf("%f",arr[1]);
    return 0;
}
```

Can you declare such function which return type is pointer to an enum?

```
#include <stdio.h>
typedef enum color{a,b,c,d,e}co;
enum color eee(){
    static co x;
    x=b+c/2;
    return x;
}

int main(){
    int num;
    num=eee();
    printf("%#d",num);
    return 0;
}

Output: 2
```

Have you worked in text video memory in c?

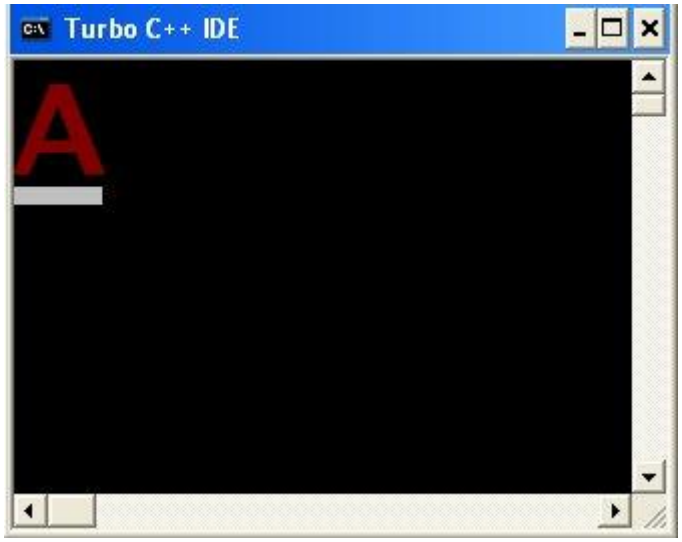
Segment number 0XB is known as text video memory. This segment is divided into 25 rows and 80 columns which creates 80*25=2000 cells. Size of each cell is two byte. Each cell is divided into two parts each of size one byte.

(a) Text byte: First byte stores character information. It stores character as ASCII code.
(b) Color byte: Second byte stores color information of text byte character.

In other word we can say each even byte stores character and each odd byte stores color. Simple example:

```
#include<stdio.h>
int main(){
int i;
char far *ptr=(char *)0XB8000000;
*ptr='A';
*(ptr+1)=4;
return 0;
}
```

Output: It will display character A in the red color as shown following screen dump:



Color scheme:

Color byte of size 8 bit stores the color information in the following manner. First four bits stores color information of ch aracter.

0000 0001: Blue color (1)

0000 0010: Green color (2)

0000 0100: Red color (4)

0000 1000: To increase the intensity of color. (8)

Note: Any other number will generate mixture of above four basic colors.

Next four bits stores color information of background of character.

0001 0000: Blue color (16)

0010 0000: Green color (32)

0100 0000: Red color (64)

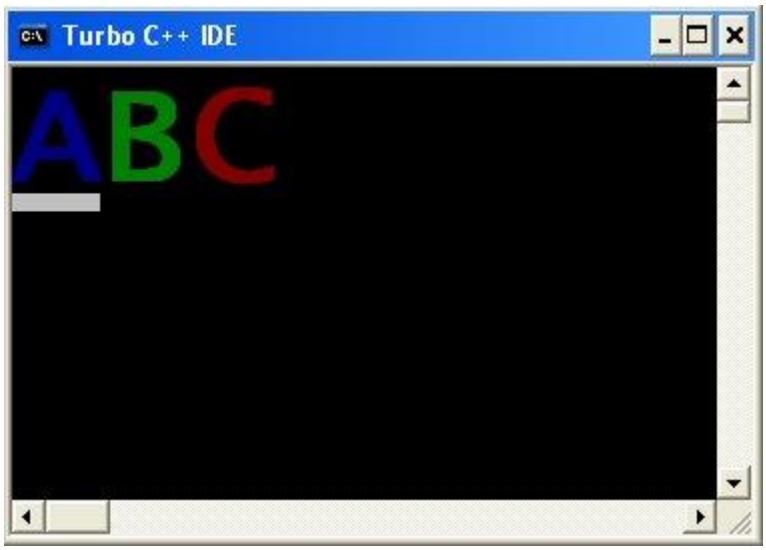
1000 0000: To increase the intensity of color. (128)

Note: Any other number will generate after mixing of above four basic colors. Examples:

(1)What will be output of following c program?

```
#include<stdio.h>
int main(){
int i;
char far *ptr=(char *)0XB8000000;
*ptr='A';
*(ptr+1)=1;
*(ptr+2)='B';
*(ptr+3)=2;
*(ptr+4)='C';
*(ptr+5)=4;
return 0;
```

```
}
Output:
```



(2)What will be output of following c program?

```
#include<stdio.h>
int main(){
int i;
char far *ptr=(char *)0XB8000000;
*ptr='W';
*(ptr+1)=1;
*(ptr+2)='O';
*(ptr+3)=2;
*(ptr+4)='R';
*(ptr+5)=4;
*(ptr+6)='L';
*(ptr+7)=1;
*(ptr+8)='D';
*(ptr+9)=2;
return 0;
}
Output:
```

(3)What will be output of following c program?

```
#include<stdio.h>
//Mixture of basic color
int main(){
int i;
char far *ptr=(char *)0XB8000000;

*ptr='W';
*(ptr+1)=3;
*(ptr+2)='O';
*(ptr+3)=5;
*(ptr+4)='R';
*(ptr+5)=6;
*(ptr+6)='L';
*(ptr+7)=7;
*(ptr+8)='D';
*(ptr+9)=3;

return 0;
}
```

Output:

(4)What will be output of following c program?

```
#include<stdio.h>
//To increase the intensity of color.
int main(){
int i;
char far *ptr=(char *)0XB8000000;
*ptr='P';
*(ptr+1)=1+8;
*(ptr+2)='O';
*(ptr+3)=2+8;
*(ptr+4)='I';
```

```
*(ptr+5)=3+8;

*(ptr+6)='N';

*(ptr+7)=4+8;

*(ptr+8)='T';

*(ptr+9)=5+8;

*(ptr+10)='E';

*(ptr+11)=6+8;

*(ptr+12)='R';

*(ptr+13)=7+8;


return 0;

}
```

Output:

(5)What will be output of following c program?

```
#include<stdio.h>

// for background color

int main(){

int i;

char far *ptr=(char *)0XB8000000;

*ptr='M';

*(ptr+1)=4+32;

*(ptr+2)='A';

*(ptr+3)=4+32;

*(ptr+4)='N';

*(ptr+5)=4+32;

*(ptr+6)='I';

*(ptr+7)=4+16;

*(ptr+8)='S';

*(ptr+9)=4+16;

*(ptr+10)='H';

*(ptr+11)=4+16;


return 0;
```

```
}
Output:
```

Can you declare pointer to structure in c?

A pointer which is pointing to a structure is known as pointer to structure. Examples of pointers to structure:

What will be output if you will execute following code?

```
#include<stdio.h>
struct address{
char *name;
char street[10];
int pin;
}cus={"S. Charvi","H-2",456003},*p=&cus;

int main(){
printf("%s %s",p->name,(*p).street);
return 0;
}
```

Output: S. Charvi H-2

Explanation:

p is pointer to structure address.-> and (*). Both are same thing. These operators are used to access data member of structure by using structure's pointer.

What is pointer to array of character in c?

A pointer to such an array which contents is character constants is known as pointer to array of character constant.

What will be output if you will execute following code?

```
#include<stdio.h>
char display(char (*)[]);
int main(){
```

```
char c;
char character[]={65,66,67,68};
char (*ptr)[]=&character;
c=display(ptr);
printf("%c",c);


return 0;
}
char display(char (*s)[]){
**s+=2;
return **s;
}
```

Output: C

Explanation: Here function display is passing pointer to array of characters and returning char data type.

```
**s+=2
=>**s=**s+2
=>**ptr=**ptr+2 //s=ptr
=>**&character= **&character+2 //ptr=&character
=>*character=*character+2 //from rule *&p =p
=>character[0]=character[0]+2 //from rule *(p+i)=p[i]
=>character [0] =67
**s=character [0] =67
```

Note: ASCII value of 'C' is 67

Can you explain pointer to two dimensional arrays in c by an example?

Examples of pointers to 2 dimensional arrays:

```
#include<stdio.h>
void main(){
long array[][3]={7l,14l,21l,28l,35l,42l};
long int (*ptr)[2][3]=&array;
printf("%li ",-0[1[0[ptr]]]);
return 0;
}
```

Output: -28

Explanation:

-0[1[0[ptr]]]

=-1[0[ptr]][0] //From rule array[i]=i[array]

=-0[ptr][1][0]

=-ptr [0] [1] [0]

=-*ptr [0] [1] //From rule array[i]=*(array+i)

=-*(&array) [0] [1]

=-(&array) [0] [1][0]

=-(*&array)[1][0] //From rule *&p=p

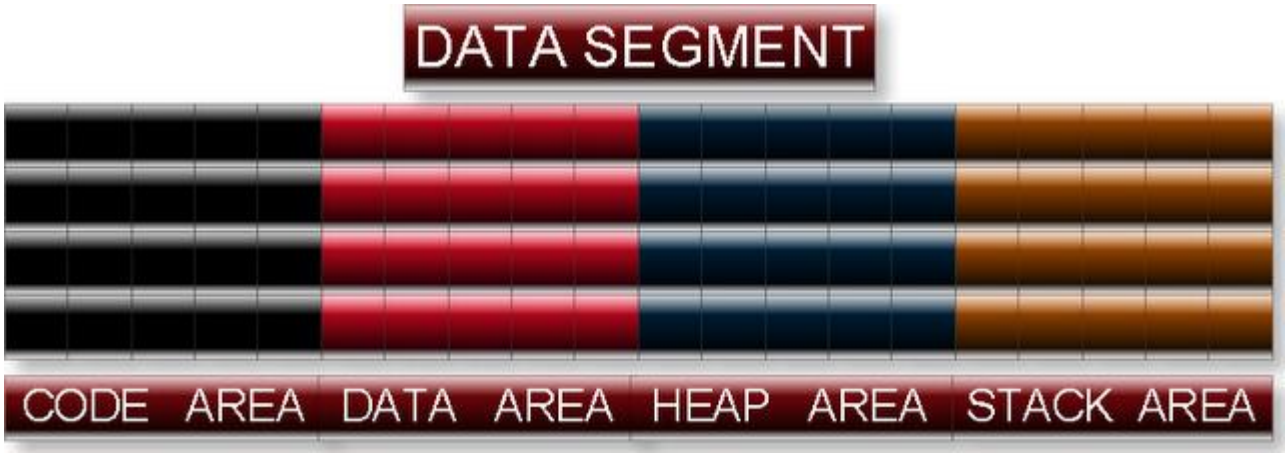
=-array[1][0]

array[1][0] means 1*(3)+ 0 = 3rd element of array starting from zero i.e. 28

What is data segment in c?

All the segments are used for specific purpose. Like segment number 15 is used for ROM, segment number 14 is used for BIOS etc.

We will discuss about how to access text video memory, graphics video memory in the pointer and union chapters of 255 bits color graphics programming. Segment number eight has special name which is known as data segment. This segment has been divided into four parts. This is very important for c programming



1. Stack area

All automatic variables and constants are stored into stack area. Automatic variables and constants in c:

1. All the local variables of default storage class.
2. Variables of storage calls auto.
3. Integer constants, character constants, string constants, float constants etc in any expression.
4. Function parameters and function return value.

Variables in the stack area are always deleted when program control reaches it out of scope. Due to this stack area is also called temporary memory area. For example:

What will be output of following c code?

```
#include<stdio.h>
int main(){
int i;
for(i=0;i<3;i++){
int a=5;
printf("%d",a);
}
return 0;
}
```

Output: 5 5 5

Explanation: Since variable a is automatic variable, it will store in the stack area. Scope of variable a is within for loop. So after each iteration variable a will be deleted from stack and in each iteration variable a will initialize. This two concepts are only for Turbo C 3.0 It follows LIFO data structure. That in the stack area of memory data is stored in last in first out. For example:

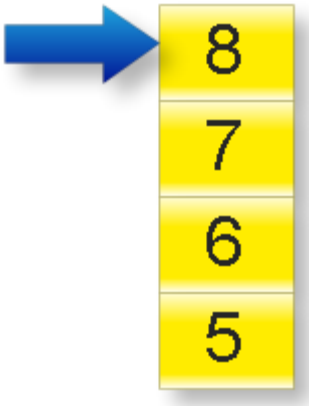
What will be output of flowing c code. (Turbo c 3.0)?

```
#include<stdio.h>
int main(){
int a =5, b = 6, c = 7,d =8;
printf("%d %d %d");
return 0;
}
```

Output: 8 7 6

Explanation:

Default storage class of variables a, b, c and d is auto .Since it automatic variable it will be sorted in the stack area of memory. It will store in the stack as



Stack always follows LIFO data structure. In the printf function, name of variables is not written explicitly. So default output will content of stack which will be in the LIFO order i.e. 8 7 6.

It has two part one for initialize variable another for non-initialize variable. All initialize variables are more nearer than not initialized variable and vice versa. For example:

What will be output of following program (Turbo c 3.0)?

```
#include<stdio.h>
int main(){
int a =5, b, c =7;
printf("%d %d %d");
return 0;
}
```

Output: 7 5 garbage value

Explanation:

Automatic variable a and c has initialized while b has not initialized. Initialize variables are more nearer than uninitialized variable .They will be stored in the stack. So due to LIFO first output will be 7 then 6 (since a is more nearer than b with respect to c) then any garbage value will be output which is present in the stack.

Note: Default storage class of any local variable is auto.

2. Data area:

All static and extern variable are stored in the data area. It is permanent memory spaAutomatic variable a and c has initialized while b has not initialized. Initialize variables are more nearer than uninitialized variable .They will be stored in the stack. So due to LIFO first output will be 7 then 6 (since a is more nearer than b with respect to c) then any garbage value will be output which is present in the stack.

Note: Default storage class of any local variable is auto.ce and variable will store in the memory unless and until program end. For example:

```
#include<stdio.h>
int main(){
int i;
for(i=0;i<3;i++){
static int a=5;
printf("%d",a);
```

```
}
return 0;
}
```

Output: 5 6 7

3. Heap area:

This memory area is use to allocate memory dynamically. In c we can allocate the memory space dynamically by using function malloc and calloc. It always allocates memory in the heap area. Its size is variable and depends upon free space in the memory.

4. Code area:

Function pointer can only access code area. Size of this area is always fixed and it is read only memory area.

Can you write a c program to check a given number is odd or even without using any conditional operators?

```
#include <stdio.h>
int fun();
int main(int argc, char *argv[]){
int num;
scanf("%d",&num);
if(num&1)
printf("odd");
else
printf("even");
return 0;
}
```

Write a c program to find out generic root of a number.

```
#include <stdio.h>
int fun();
int main(int argc, char *argv[]){
int num,x;
scanf("%d",&num);
printf("%d",(x=num%9)?x:9);
return 0;
}
```

Can you write a function which executes after the main function in c?

```
#include <stdio.h>
#include<stdlib.h>

void fun();
```

```
int main(int argc, char *argv[]){
   atexit(fun);
   printf("cquestion");
   return 0;
}
void fun(){
  printf("bank");
}
```

What is a multilevel pointer in c?

A pointer is pointer to another pointer which can be pointer to others pointers and so on is known as multilevel pointers. We can have any level of pointers. Examples of multilevel pointers in c:

(1) What will be output if you will execute following code?

```
#include<stdio.h>
int main(){
int s=2,*r=&s,**q=&r,***p=&q;
printf("%d",p[0][0][0]);
return 0;
}
```

Output: 2

Explanation:

As we know p[i] =*(p+i)

So,

P[0][0][0]=*(p[0][0]+0)=**p[0]=***p

Another rule is: *&i=i

So,

p= (&q) =**q=** (&r) =*r=*(&s) =s=2

(2) What will be output if you will execute following code?

```
#include<stdio.h>
#define int int*
int main(){
int *p,q;
p=(int *)5;
q=10;
printf("%d",q+p);
return 0;
}
```

Output: 25

Explanation: If you will see intermediate file you will find following code:

```
#include<stdio.h>
void main(){
int **p,q;
```

```
p=(int **)5;
q=10;
printf("%d",q+p);
return 0;
}
```

Explanations:

Here q pointer and p is a number.

In c, Address + number = Address

So, New address = old address + number * Size of data type to which pointer is pointing.

= 5 + 10 * sizeof (*int)

= 5+10*2 = 25.

Note: We are assuming default pointer is near. Actually it depends upon memory model.

What is function recursion in c programming? Can you write any program using function?

Calling of same function from its function body is known as function recursion. It is alternative of loop. Any c program which is possible using loop it must be possible using function recursion. Simple example:

Find the sum of all even numbers from 0 to 20 using function recursion. Program:

```
#include<stdio.h>
int main(){
int total;
total=sum(2);
printf("%d",total);
return 0;
}
int sum(int i){
static int even=0;
if(i<=20){
even=even+i;
sum(i+2); //calling same function
}
return even;
}
```

Output:

It is very difficult to understand the execution as well as to write the function recursion program.

How to write function recursion program in easier way:

I have already told it is very difficult to write function recursion program directly. If any person is writing such program directly he may be memorized that program. I am telling a very nice trick: how to write function recursion program in easier way?

As we know any c program which possible using loop it must be possible using function recursion.

Steps for how to write function recursion program:

Step1: First of all write same program using while loop and function. (Except main function)

Step 2: In that function make all local variable static.

Step 3: Replace while keyword by if.

Step 4: The increment or decrement of variable which is used for condition checking, replace with function call and pass the parameter of that

incremented or decremented variable. Now understand by example:

Find the sum of all even numbers from 0 to 20 using function recursion.

Step 1: Write the same program using while loop and function. Here function is sum.

```
#include<stdio.h>
int main(){
int total;
total=sum(2);
printf("%d",total);
return 0;
}
int sum(int i){
int even=0;
while(i<=20){
even=even+i;
i=i+2;
}
return even;
}
```

Step 2: Make local variable even as static variable

```
int sum(int i){
static int even=0;
while(i<=20){
even=even+i;
i=i+2;
}
return even;
}
```

Step 3: Replace while keyword by if keyword.

```
int sum(int i){
int even=0;
if(i<=20){
even=even+i;
i=i+2;
}
return even;
}
```

Step 4: Since here variable i has used in condition checking. So replace the statement i=i+2 by sum (i+2).

```
int sum(int i){
int even=0;
if(i<=20){
even=even+i;
sum(i+2);
}
```

```
return even;
}
```

Following only three simple change you can write any difficult function recursion program.

```
#include<stdio.h>
int main(){
int total;
total=sum(2);
printf("%d",total);
return 0;
}
int sum(int i){
int even=0;
if(i<=20){
even=even+i;
sum(i+2);
}
return even;
}
```

One more example:

Write a program to find a factorial of given number using function recursion.

Step 1: write same program using while loop and function.

```
#include<stdio.h>
int main(){
int fact,num;
scanf("%d",&num);
fact=factorial(num);
printf("%d",fact);
return 0;
}
int factorial(int num){
int fact=1; //make it static
while(num>0){ //replace while by if
fact=fact*num;
num--; // replace by function call as factorial(num-1);
}
return fact;
}
```

After following step1, step 2, step 3:

```
#include<stdio.h>
int main(){
int fact,num;
scanf("%d",&num);
fact=factorial(num);
```

```
printf("%d",fact);
return 0;
}
int factorial(int num){
static int fact=1;
if(num>0){
fact=fact*num;
factorial (num-1);
}
return fact;
}
```

Note1: In step 3 while calling function don't pass the parameter using unary operator like factorial (num--);
Note2: write the function in such a way parameter of calling function must be used in conditional statement. For example in the above program:

int factorial(int num)

num is function parameter and num is also used in the conditional if statement.

What is data structure?

A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

List out the areas in which data structures are applied extensively?

1. Compiler Design,
2. Operating System,
3. Database Management System,
4. Statistical analysis package,
5. Numerical Analysis,
6. Graphics,
7. Artificial Intelligence,
8. Simulation

What are the major data structures used in the following areas : RDBMS, Network data model and Hierarchical data model.

1. RDBMS = Array (i.e. Array of structures)
2. Network data model = Graph
3. Hierarchical data model = Trees

If you are using C language to implement the heterogeneous linked list, what pointer type will you use?

The heterogeneous linked list contains different data types in its nodes and we need a link, pointer to connect them. It is not possible to use ordinary pointers for this. So we go for void pointer. Void pointer is capable of storing pointer to any type as it is a generic pointer type.

Minimum number of queues needed to implement the priority queue?

Two. One queue is used for actual storing of data and another for storing priorities

---

What is the data structures used to perform recursion?

Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls.

Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

What are the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?

Polish and Reverse Polish notations.

Convert the expression ((A + B) * C - (D - E) ^ (F + G)) to equivalent Prefix and Postfix notations.

1. **Prefix Notation:** - * +ABC ^ - DE + FG
2. **Postfix Notation:** AB + C * DE - FG + ^ -

Sorting is not possible by using which of the following methods? (Insertion, Selection, Exchange, Deletion

Sorting is not possible in Deletion. Using insertion we can perform insertion sort, using selection we can perform selection sort, using exchange we can perform the bubble sort (and other similar sorting methods). But no sorting method can be done just using deletion.

What are the methods available in storing sequential files ?

1. Straight merging,
2. Natural merging,
3. Polyphase sort,
4. Distribution of Initial runs.

List out few of the Application of tree data-structure?

1. The manipulation of Arithmetic expression,
2. Symbol Table construction,
3. Syntax analysis

List out few of the applications that make use of Multilinked Structures?

1. Sparse matrix,
2. Index generation

In tree construction which is the suitable efficient data structure? (Array, Linked list, Stack, Queue)

Linked list is the suitable efficient data structure.

What is the type of the algorithm used in solving the 8 Queens problem?

Backtracking

In an AVL tree, at what condition the balancing is to be done?

If the 'pivotal value' (or the 'Height factor') is greater than 1 or less than -1.

What is the bucket size, when the overlapping and collision occur at same time?

One. If there is only one entry possible in the bucket, when the collision occurs, there is no way to accommodate the colliding value. This results in the overlapping of values.

Classify the Hashing Functions based on the various methods by which the key value is found.

1. Direct method,
2. Subtraction method,
3. Modulo-Division method,
4. Digit-Extraction method,
5. Mid-Square method,
6. Folding method,

7. Pseudo-random method.

What are the types of Collision Resolution Techniques and the methods used in each of the type?

1. **Open addressing (closed hashing),** The methods used include: Overflow block.
2. **Closed addressing (open hashing),** The methods used include: Linked list, Binary tree.

In RDBMS, what is the efficient data structure used in the internal storage representation?

B+ tree. Because in B+ tree, all the data is stored only in leaf nodes, that makes searching easier. This corresponds to the records that shall be stored in leaf nodes.

What is a spanning Tree?

A spanning tree is a tree associated with a network. All the nodes of the graph appear on the tree once. A minimum spanning tree is a spanning tree organized so that the total edge weight between nodes is minimized.

Does the minimum spanning tree of a graph give the shortest distance between any 2 specified nodes?

No. The Minimal spanning tree assures that the total weight of the tree is kept at its minimum. But it doesn't mean that the distance between any two nodes involved in the minimum-spanning tree is minimum.

Which is the simplest file structure? (Sequential, Indexed, Random)

Sequential is the simplest file structure.

Whether Linked List is linear or Non-linear data structure?

According to Access strategies Linked list is a linear one. According to Storage Linked List is a Non-linear one.

When I can call it complete Binary Tree?

It is said to be complete when each node that has a right child also has a left child. Having a left child does not require a node to have a right child. Alternate Binary tree is a tree where their is always a left node for right node but may be may not be right node for left node.

What is AVL tree?

AVL tree is self balancing tree, in which balancing factor lie between the -1 to 1.

How can I find the number of possible tree in the given tree?

Number of possible tree = (2 power of n) – n

What is Hashing?

Hashing is a technique to retrieve records from memory quickly.

What is difference between linear and non linear data structures?

Linear data structures are data structures whose data are in linear. eg: Arrays

Non linear are data structures whose data are non linear format. eg: Trees

---

List the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?

Polish and Reverse Polish notations.

Can I use Selection method to for sorting?

Yes, Selection method is used for Selection sort.

What is the main difference between Storage structure and file structure and how?

The representation of a particular data structure in the memory of a computer is called storage structure whereas a storage structure representation in auxiliary memory is often called a file structure.

Clarify whether Linked List is linear or Non-linear data structure?

Link list is always linear data structure because every element (NODE) having unique position and also every element has its unique successor and predecessor. Also,linear collection of data items called nodes and the linear order is given by means of pointers. Each node is divided into two parts. First part contains information of the element and another part contains the address of the next node in the list.

Explain simulation

Simulation is the process of forming an abstract model from a real situation in order to understand the impact of modifications and the effect of introducing various strategies on the situation. It is a representation of real life system by another system, which describes the important characteristics of real system and allows experiments on it.

Does the minimum spanning tree of a graph give the shortest distance between any 2 specified nodes?

Minimal spanning tree assures that the total weight of the tree is kept at its minimum but it doesn't mean that the distance between any two nodes involved in the minimum-spanning tree is minimum

---

In an AVL tree, at what condition the balancing is to be done?

If the pivotal value (or the Height factor) is greater than 1 or less than 1.If the balance factor of any node is other than 0 or 1 or -1 then balancing is done.The balancing factor is height. The difference in height of the right subtree and right subtree should be +1 , -1 or 0

What is the main difference between ARRAY and STACK?

Stack follows LIFO. Thus the item that is first entered would be the last removed.In array the items can be entered or removed in any order. Basically each member access is done using index and no strict order is to be followed here to remove a particular element .Array may be multi dimensional or one dimensional but stack should be one-dimensional.

Size of array is fixed, while stack can be grow or shrink. We can say stack is dynamic data structure.