Biyani's Group of Girls' Colleges

B.C.A. I YEAR

Principles of Programming Language(Through C)

PRE-UNIVERSITY EXAM 2018(Set 1)

MAX MARKS :100

PART - I

A- Attempt all the questions (very short answer). Your answers should not exceed the maximum word limit of 40 for each question. Each question carries 2 marks.

1. What is variable?
   A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

2. What is scanf?
   scanf is a function that reads data with specified format from a given string stream source, originated from C programming language, and is present in many other programming languages.
   The scanf function prototype is:

3. Explain the function to take user input?
   scanf is a function that reads data with specified format from a given string stream source, originated from C programming language, and is present in many other programming languages.

4. What is difference between (=) and (==) ?

   Assignment Operator (=)
   = is an **Assignment Operator** in C, C++ and other programming languages, It is **Binary Operator** which operates on two operands. = assigns the value of right side expression's or variable's value to the left side variable.

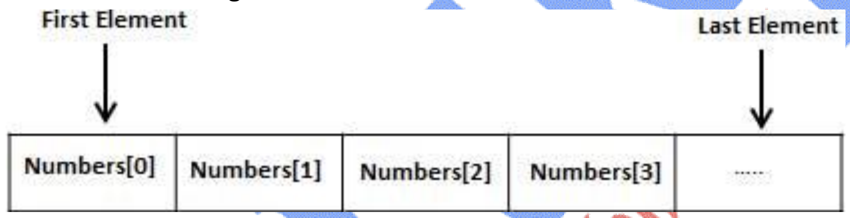
   Equal To Operator (==)
   == is an Equal To Operator in C and C++ only, It is Binary Operator which operates on two operands.== compares value of left and side expressions, return 1 if they are equal other will it will return 0.

5. What do you understand by while(1234) ?

while loop in C programming repeatedly executes a target statement as long as a given condition is true.
Here condition is a non zero value so it is always true. Hence it is an infinite loop.

6. What is looping?
A loop statement allows us to execute a statement or group of statements multiple times. a loop is a sequence of instruction s that is continually repeated until a certain condition is reached.

7. What is array ?
Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
ll arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



8. What is STDIO.H?
The C programming language provides many standard library functions for file input and output. These functions make up the bulk of the C standard library header <stdio.h>.
The **stdio.h** header defines three variable types, several macros, and various functions for performing input and output

9. What is pre-processor directive?
The **C Preprocessor** is not a part of the compiler, but is a separate step in the compilation process. In simple terms, a C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation
All preprocessor commands begin with a hash symbol (#). It must be the first nonblank character, and for readability, a preprocessor directive should begin in the first column.
Examples : **#define, #include**

10. What is function declaration?
A function is a group of statements that together perform a task. A function declaration tells the compiler about a function's name, return type, and parameters.
A function declaration has the following parts –

return_type function_name(parameter list);
Example
int max(int num1, int num2)

PART- II

B- Attempt all the questions (short answers). Your answer should not exceed the maximum word limit of 80 words for each question. Each question carriers 4 marks.

1. Write short note on programming languages.
   A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.

- **Machine languages**, that are interpreted directly in hardware
- **Assembly languages**, that are thin wrappers over a corresponding machine language
- **High-level languages**, that are anything machine-independent

2. Explain arithmetic operators?

The Arithmetic operators are some of the C Programming Operator, which are used to perform arithmetic operations includes operators like Addition, Subtraction, Multiplication, Division and Modulus. All these operators are binary operators which means they operate on two operands.

Arithmetic Operators in C	Operation	Example
+	Addition	10 + 2 = 12
–	Subtraction	10 – 2 = 8
*	Multiplication	10 * 2 = 20
/	Division	10 / 2 = 5
%	Modulus – It returns the remainder after the division	10 % 2 = 0 (Here remainder is zero). If it is 10 % 3 then it will be 1.

3. Explain For loop with an example?
   A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
   Syntax
   The syntax of a for loop in C programming language is –

```
for ( init; condition; increment ) {
  statement(s);
}
```

Example:

```
#include <stdio.h>
int main()
{
  int i;
  for (i=1; i<=3; i++)
  {
     printf("%d\n", i);
  }
  return 0;
}
```

Output

```
1
2
3
```

4. Explain various types of function?
   A function is a block of code that performs a specific task.
   Types of functions in C programming
   Depending on whether a function is defined by the user or already included in C compilers, there are two types of functions in C programming
   There are two types of functions in C programming:

   - Standard library functions
   - User defined functions

Standard library functions

The standard library functions are built-in functions in C programming to handle tasks such as mathematical computations, I/O processing, string handling etc.
These functions are defined in the header file. When you include the header file, these functions are available for use. For example:
The printf() is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in "stdio.h" header file.
There are other numerous library functions defined under "stdio.h", such as scanf(), fprintf(), getchar() etc. Once you include "stdio.h" in your program, all these functions are available for use.

User-defined functions

As mentioned earlier, C allow programmers to define functions. Such functions created by the user are called user-defined functions.

Depending upon the complexity and requirement of the program, you can create as many user-defined functions as you want.

5. What is pointers? Explain with an example.

A pointer is a variable which contains the address in memory of another variable. We can have a pointer to any variable type.
The ***unary*** or ***monadic*** operator **&** gives the ``address of a variable''.
The ***indirection*** or dereference operator ***** gives the ``contents of an object ***pointed to*** by a pointer''.
To declare a pointer to a variable do:

```
int *pointer;
```

NOTE: We must associate a pointer to a particular type: You can't assign the address of a **short int** to a **long int**, for instance.
Consider the effect of the following code:

```
int x = 1, y = 2;
                     int *ip;

                     ip = &x;
```

PART- III

C- Attempt all the questions (long answer). Draw neat and comprehensive sketches wherever necessary to clearly illustrate your answer. Each question carries 12 marks.

1. What is Programming Paradigm? Explain each of them in detail.

The function of a computer system are control by computer program. A computer program is a clear step by step finit set of instruction. A computer program is written in a computer language called programming language.

A programming language must be a universal.

A programming language must be implement table on a computer.

The different types of programming language:-

1 MACHINE LEVEL LANGUAGE

2 ASSAMBLY LEVEL LENGUAGE

3 HIGH LEVEL LENGUAGE

1. Machine level language:- It consist a sequence of zeros and ones each kind of CPU has its all called machine language. It is fast and efficient . No translation is required for this language. But it is fortable and not programmer friendly.

2. Assambly level language:- Assembly level language use mnemonics to represent machine instruction.

Eg:- Add 2,5

Mov x,b

Slover than machine language but still much faster than high level language and assembly level language called law lavel language.

3. High level language:- the program in high level language are similar to the English type language with some syntax and symentix of a particular programming language. It has a set of built in language primitive and grametical rules. It always need a transferable.

language/paradigm programming:-

Programming paradigm are our most basic tools and be throughly master than to use them effectively. A programming paradigm is an approach to solve programming problems.

TYPES OF LANGUAGE PARADIGM:-

1. Impetrative:- this paradigm program is a series of statements containing variables And program contain of the computer continuslly.

2. Object oriented:- In this paradigm all the programming language are the base on object communication. Object communication is done by the sending messages to each other.

Eg:-java, c#(c sharp),small talk.

3.Functional:- This paradigm consist of various function programs execution is done by function calling and returning results.

Eg:-ml, Haskell ,miranda^{tm}

4.Logical oriented language:-it consist of a set of pridicates and rules of inference predicates are the statement of facts and rules of inference are the conditional statement.

Eg:-PROLOG

5.Scripting language:-by the presence of very high level language features.

Eg:-java script

Eg:-vb script

6.Concrent process:-in this paradigm work is done by the use of concurrent process.

Eg:-PL/I,ALGOL,ADA,MODULa

History of programming language:-

1.Fortran:- it introduce simonix expression and array and also procedurers.

2.Cobal:-it include the concept of data description.

3.Algol 60:-it is design for communicating. It introduce the concept of block structure.

FORTRN and ALGOL 60 are most usefull for numeric complication.

4.PL/I:-it must the features of all three language to design genral purpose programming language.

5. PASCAL:-it is imperative procedural language.

6.small talk:-it is an object oriented dynamically type reflected programming language.

7. LISP:-it is suitable for artificial intelligence program.

8.PROLOG:-it is based on logical programming.

9. ADA:-it is structured imperative and object oriented high level programming language.

10. ML:-ml is a genral purpose functional programming language. It has been cherecterised as LISP with types.

11.HASKELL:-haskell is a standerised genral purpose purely functional programming language.

12.C++:-it is a genral purpose programming language. It has imperative,object oriented & generic programming feature which also provide facilities for low level memory manuplation.

13.JAVA:-purelly object oriented programming language with the features including exception handing and thread.

14.C#(c sharp):-object oriented programming language that aims to combine the computing power of c++ with the programming case of VB(visited bask).

Syntex and symentix:-
Syntax:- it concerned with the form of program expression , command & other concerned that must be arrange to make well formed program.

Symentix:-it is concerned with the meaning of the program. How a well promote program may be expected to behave when executed on a computer.

2. Explain Operators and its types.

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators –

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators

Arithmetic Operators

The Arithmetic operators are some of the C Programming Operator, which are used to perform arithmetic operations includes operators like Addition, Subtraction, Multiplication, Division and Modulus. All these operators are binary operators which means they operate on two operands.

Arithmetic Operators in C	Operation	Example
+	Addition	10 + 2 = 12
–	Subtraction	10 – 2 = 8
*	Multiplication	10 * 2 = 20
/	Division	10 / 2 = 5
%	Modulus – It returns the remainder after the division	10 % 2 = 0 (Here remainder is zero). If it is 10 % 3 then it will be 1.

Relational Operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not

		true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

Logical Operators

Following table shows all the logical operators supported by C language. Assume variable **A** holds 1 and variable **B** holds 0, then −

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

Bitwise Operators

Bitwise operator works on bits and perform bit-by-bit operation. The truth tables for &, |, and ^ is as follows −

p	Q	p & q	p | q	p ^ q

0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Assume A = 60 and B = 13 in binary format, they will be as follows −

A = 0011 1100

B = 0000 1101

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

~A = 1100 0011

The following table lists the bitwise operators supported by C. Assume variable 'A' holds 60 and variable 'B' holds 13, then −

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011

		0001

Assignment Operators

The following table lists the assignment operators supported by the C language −

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A

Misc Operators ↦ sizeof & ternary

Besides the operators discussed above, there are a few other important operators including sizeof and ? : supported by the C Language.

Operator	Description	Example
sizeof()	Returns the size of a variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of a variable.	&a; returns the actual address of the variable.

*	Pointer to a variable.	*a;
? :	Conditional Expression.	If Condition is true ? then value X : otherwise value Y

Operators Precedence in C

Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.

For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right

Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

3. What is array? Explain types of array each with suitable example.

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows –

```
type arrayName [ arraySize ];
```

This is called a single-dimensional array. The arraySize must be an integer constant greater than zero and type can be any valid C data type. For example, to declare a 10-element array called balance of type double, use this statement –

```
double balance[10];
```

Here balance is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

You can initialize an array in C either one by one or using a single statement as follows –

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

```
double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array −

```
balance[4] = 50.0;
```

The above statement assigns the 5th element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1.

Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example −

```
double salary = balance[9];
```

The above statement will take the 10th element from the array and assign the value to salary variable. The following example Shows how to use all the three above mentioned concepts viz. declaration, assignment, and accessing arrays −

```
#include <stdio.h>

int main () {

  int n[ 10 ]; /* n is an array of 10 integers */

  int i,j;

  /* initialize elements of array n to 0 */

  for ( i = 0; i < 10; i++ ) {

    n[ i ] = i + 100; } /* set element at location i to i + 100 */

  /* output each array element's value */

  for (j = 0; j < 10; j++ ) {

    printf("Element[%d] = %d\n", j, n[j] );

  }   return 0;

}
```

When the above code is compiled and executed, it produces the following result −

```
Element[0] = 100

Element[1] = 101

Element[2] = 102

Element[3] = 103

Element[4] = 104

Element[5] = 105

Element[6] = 106

Element[7] = 107

Element[8] = 108

Element[9] = 109
```

4. Write program to print fabonic series with and without fuction .

Fibonacci Series in C

Fibonacci Series in C: In case of fibonacci series, *next number is the sum of previous two numbers* for example 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. The first two numbers of fibonacci series are 0 and 1.

There are two ways to write the fibonacci series program:

- Fibonacci Series without recursion
- Fibonacci Series using recursion

Fibonacci Series in C without recursion

```
        #include<stdio.h>

    int main()

    {

     int n1=0,n2=1,n3,i,number;

     printf("Enter the number of elements:");

     scanf("%d",&number);
```

```
        printf("\n%d %d",n1,n2);//printing 0 and 1
        for(i=2;i<number;++i)//loop starts from 2 because 0 and 1 are already printed
        {
         n3=n1+n2;
         printf(" %d",n3);
         n1=n2;
         n2=n3;
        }
         return 0;
        }
```

Output:

```
Enter the number of elements:15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

<u>Fibonacci Series using recursion in C</u>

```
 #include<stdio.h>
void printFibonacci(int n){
   static int n1=0,n2=1,n3;
   if(n>0){
        n3 = n1 + n2;
        n1 = n2;
        n2 = n3;
        printf("%d ",n3);
        printFibonacci(n-1);
   }
}
int main(){
   int n;
```

```
    printf("Enter the number of elements: ");

    scanf("%d",&n);

    printf("Fibonacci Series: ");

    printf("%d %d ",0,1);

    printFibonacci(n-2);//n-2 because 2 numbers are already printed

   return 0;

  }
```

Output:

```
Enter the number of elements:15

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

6. What is recursion? Explain with an example.

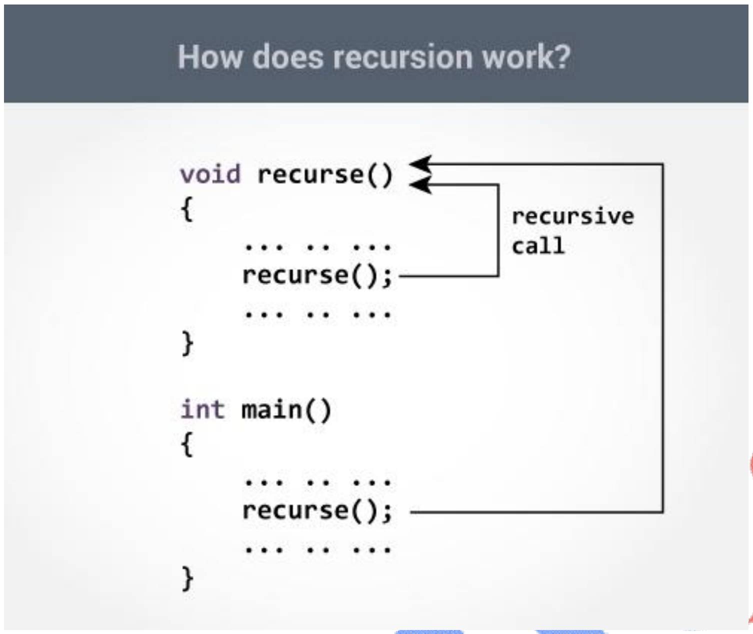
   Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.

   ```
   void recursion() {
      recursion(); /* function calls itself */
   }

   int main() {
      recursion();
   }
   ```

   The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

   Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.



```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```

Example : Number Factorial

The following example calculates the factorial of a given number using a recursive function –

```
#include <stdio.h>
unsigned long long int factorial(unsigned int i) {
  if(i <= 1) {
    return 1;
  }
  return i * factorial(i - 1);
}
int  main() {
  int i = 12;
  printf("Factorial of %d is %d\n", i, factorial(i));
  return 0;
}
```

When the above code is compiled and executed, it produces the following result –

```
Factorial of 12 is 479001600
```

Example 2 : Sum of Natural Numbers Using Recursion

```
#include <stdio.h>
int sum(int n);
int main()
{
   int number, result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum=%d", result);
}
int sum(int num)
{
   if (num!=0)
      return num + sum(num-1); // sum() function calls itself
   else
      return num;
}
```

Output

```
Enter a positive integer:
3
6
```