UNIT I
Part - A

1. What are Keywords?
   Keywords are certain reserved words that have standard and pre-defined
   meaning in ‘C’. These keywords can be used only for their intended purpose.
   Example: float, int, break, continue, goto, if, if- else, while, do-while, etc

2. What is the difference between if and while statement?

<table>
<thead>
<tr>
<th></th>
<th>if</th>
<th>while</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>It is a conditional statement</td>
<td>(i) It is a loop control statement</td>
</tr>
<tr>
<td>(ii)</td>
<td>If the condition is true, it executes some statements.</td>
<td>(ii) Executes the statements within the while block if the condition is true.</td>
</tr>
<tr>
<td>(iii)</td>
<td>If the condition is false then it stops the execution the statements.</td>
<td>(iii) If the condition is false the control is transferred to the next statement of the loop.</td>
</tr>
</tbody>
</table>

3. What is the difference between while loop and do…while loop?
   In the while loop the condition is first executed. If the condition is true then it executes the body of the loop. When the condition is false it comes of the loop. In the do…while loop first the statement is executed and then the condition is checked. The do…while loop will execute at least one time even though the condition is false at the very first time.

4. What will happen when you access the array more than its dimension?
   When you access the array more than its dimensions some garbage value is stored in the array.

5. Write the limitations of getchar( ) and scanf( ) functions for reading strings
   getchar( )
   To read a single character from stdin, then getchar() is the appropriate.
   Scanf()
   scanf( ) allows to read more than just a single character at a time.
6. What is the output of the programs given below?

main() {
    float a;
    int x=6, y=4;
    a=x\y;
    printf(“Value of a=%f”, a);
}

Output:
1.

main() {
    float a;
    int x=6, y=4;
    a=(float) x/y;
    printf(“Value of a=%f”, a);
}

Output:
1.500000

7. Distinguish between Call by value Call by reference.

<table>
<thead>
<tr>
<th>Call by value</th>
<th>Call by reference.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) In call by value, the value of actual agreements is passed to the formal arguments and the operation is done on formal arguments.</td>
<td>a) In call by reference, the address of actual argument values is passed to formal argument values.</td>
</tr>
<tr>
<td>b) Formal arguments values are photocopies of actual arguments values.</td>
<td>b) Formal arguments values are pointers to the actual argument values.</td>
</tr>
<tr>
<td>c) Changes made in formal arguments valued do not affect the actual arguments values.</td>
<td>c) Since Address is passed, the changes made in the both arguments values are permanent.</td>
</tr>
</tbody>
</table>

8. What is the difference between while(a) and while(!a)?
while(a) means while(a!=0)
while(!a) means while(a==0)
9. Why we don’t use the symbol ‘&’ symbol, while reading a String through scanf()?

The ‘&’ is not used in scanf() while reading string, because the character variable itself specifies as a base address.

Example: name, &name[0] both the declarations are same.

10. What is the difference between static and auto storage classes?

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>Memory</td>
<td>Memory</td>
</tr>
<tr>
<td>Initial value</td>
<td>Zero</td>
<td>Garbage value</td>
</tr>
<tr>
<td>Scope</td>
<td>Local to the block in which the variables is defined</td>
<td>Local to the block in which the variable is defined.</td>
</tr>
<tr>
<td>Life</td>
<td>Value of the variable persists between different function calls.</td>
<td>The block in which the variable is defined.</td>
</tr>
</tbody>
</table>

11. What is the output of the program?

```c
main()
{
    increment();
    increment();
    increment();
}

increment()
{
    static int i=1;
    printf("%d\n",i)
    i=i+1;
}
OUTPUT:
1 2 3
```

12. Define delimiters in ‘C’.

<table>
<thead>
<tr>
<th>Delimiters</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>Colon</td>
</tr>
<tr>
<td>;</td>
<td>Semicolon</td>
</tr>
<tr>
<td>( )</td>
<td>Parenthesis</td>
</tr>
<tr>
<td>[ ]</td>
<td>Square Bracket</td>
</tr>
<tr>
<td>{ }</td>
<td>Curly Brace</td>
</tr>
<tr>
<td>#</td>
<td>Hash</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S No</th>
<th>break</th>
<th>continue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exits from current block / loop</td>
<td>Loop takes next iteration</td>
</tr>
<tr>
<td>2</td>
<td>Control passes to next statement</td>
<td>Control passes to beginning of loop</td>
</tr>
<tr>
<td>3</td>
<td>Terminates the program</td>
<td>Never terminates the program</td>
</tr>
</tbody>
</table>

14. List the types of operators.

<table>
<thead>
<tr>
<th>S No</th>
<th>Operators Types</th>
<th>Symbolic Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arithmetic operators</td>
<td>=, -, *, / and %</td>
</tr>
<tr>
<td>2</td>
<td>Relational operators</td>
<td>&gt;, &lt;, ==, &gt;=, &lt;= and !=</td>
</tr>
<tr>
<td>3</td>
<td>Logical operators</td>
<td>&amp;&amp;,</td>
</tr>
<tr>
<td>4</td>
<td>Increment and Decrement</td>
<td>++, --</td>
</tr>
<tr>
<td>5</td>
<td>operators</td>
<td>=, +=, -=, * =, / =, ^=, ^=, ^=, &amp;=,</td>
</tr>
<tr>
<td>6</td>
<td>Assignment</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>operators Bitwise</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>operators Comm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>operator Conditional</td>
<td></td>
</tr>
<tr>
<td></td>
<td>operator</td>
<td></td>
</tr>
</tbody>
</table>

15. Distinguish between while..do and do..while statement in C.

While                               

(i) Executes the statements within the while block if only the condition is true.

DO..while                           

(i) Executes the statements within the while block at least once.
(ii) The condition is checked at the starting of the loop

(ii) The condition is checked at the end of the loop

16. Compare switch() and nested if statement.

<table>
<thead>
<tr>
<th>S No</th>
<th>switch() case</th>
<th>nested if</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test for equality ie., only constant values are applicable.</td>
<td>It can equate relational (or) logical expressions.</td>
</tr>
<tr>
<td>2</td>
<td>No two case statements in same switch.</td>
<td>Same conditions may be repeated for a number of times.</td>
</tr>
<tr>
<td>3</td>
<td>Character constants are automatically converted to integers.</td>
<td>Character constants are automatically converted to integers.</td>
</tr>
<tr>
<td>4</td>
<td>In switch() case statement nested if can be used.</td>
<td>In nested if statement switch case can be used.</td>
</tr>
</tbody>
</table>

17. Give the syntax for the ‘for’ loop statement

```c
for (Initialize counter; Test condition; Increment / Decrement)
{
    statements;
}
```

- **Initialization counter** sets the loop to an initial value. This statement is executed only once.
- The **test condition** is a relational expression that determines the number of iterations desired or it determines when to exit from the loop.
- The **increment / decrement parameter** decides how to make changes in the loop.
18. What is the use of sizeof() operator?
The sizeof () operator gives the bytes occupied by a variable. No of bytes occupied varies from variable to variable depending upon its data types.

Example:
```c
int x, y;
printf("%d", sizeof(x));
```

Output:
```
2
```

19. What is a loop control statement?
Many tasks done with the help of a computer are repetitive in nature. Such tasks can be done with loop control statements.

20. What are global variable in ‘C’?
- This section declares some variables that are used in more than one function. such variable are called as global variables.
- It should be declared outside all functions.

21. Write a program to swap the values of two variables (without temporary variable).
```c
#include <stdio.h>
#include <conio.h>
void main()
{
    int a =5; b = 10;
    clrscr( );
    printf("Before swapping a = %d b = %d \", a , b);
    a = a + b;
    b = a – b;
    a = a – b;
    printf("After swapping a = %d b = %d\", a,b);
    getch( );
}
```

Output:
```
Before swapping a = 5 b = 10
After swapping a = 10 b = 5
```

22. What is an array?
An array is a group of similar data types grouped under a common name.

```c
int a[10];
```
Here a[10] is an array with 10 values.

23. **What is a Pointer? How a variable is declared to the pointer?**

Pointer is a variable which holds the address of another variable.

**Pointer Declaration:**

```c
datatype *variable-name;
```

**Example:**

```c
int *x, c=5;
x=&a;
```

24. **What are the uses of Pointers?**

- Pointers are used to return more than one value to the function
- Pointers are more efficient in handling the data in arrays
- Pointers reduce the length and complexity of the program
- They increase the execution speed
- The pointers saves data storage space in memory

25. **What is the output of the program?**

```c
main() junk(int i, int j)
{
    int i=5;j=2;
    junk(i,j); printf(“n %d %d”,i,j);
}
```

**Output:**

1. 2
2.

26. **What are * and & operators means?**

- `*` operator means ‘value at the address’
- `&` operator means ‘address of’

27. **What is meant by Preprocessor?**

Preprocessor is the program, that process our source program before the compilation.
28. **How can you return more than one value from a function?**
   A Function returns only one value. By using pointer we can return more than one value.

29. **What are the main elements of an array declaration?**
   - Array name
   - Datatype
   - Size

30. **List the header files in ‘C’ language.**
   - `<stdio.h>` contains standard I/O functions
   - `<ctype.h>` contains character handling functions
   - `<stdlib.h>` contains general utility functions
   - `<string.h>` contains string manipulation functions
   - `<math.h>` contains mathematical functions
   - `<time.h>` contains time manipulation functions

31. **What are the pre-processor directives?**
   - Macro Inclusion
   - Conditional Inclusion
   - File Inclusion

32. **What is the difference between an array and pointer?**
   Difference between arrays and pointers are as follows.

<table>
<thead>
<tr>
<th>Array</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Array allocates space automatically.</td>
<td>1. Pointer is explicitly assigned to point to</td>
</tr>
<tr>
<td>2. It cannot be resized.</td>
<td>an allocated space.</td>
</tr>
<tr>
<td>3. It cannot be reassigned.</td>
<td>2. It can be resized using realloc ().</td>
</tr>
<tr>
<td>4. Size of(array name) gives the number of</td>
<td>3. Pointers can be reassigned.</td>
</tr>
<tr>
<td>bytes occupied by the array.</td>
<td>4. Sizeof(pointer name) returns the number</td>
</tr>
<tr>
<td></td>
<td>of bytes used to store the pointer variable.</td>
</tr>
</tbody>
</table>

www.studentsfocus.com
33. Is it better to use a macro or a function?
Macros are more efficient (and faster) than function, because their corresponding code is inserted directly at the point where the macro is called. There is no overhead involved in using a macro like there is in placing a call to a function.
However, macros are generally small and cannot handle large, complex coding constructs. In cases where large, complex constructs are to handled, functions are more suited, additionally; macros are expanded inline, which means that the code is replicated for each occurrence of a macro.

34. List the characteristics of Arrays.

- All elements of an array share the same name, and they are distinguished from one another with help of an element number.
- Any particular element of an array can be modified separately without disturbing other elements.

35. What are the types of Arrays?
1. One-Dimensional Array
2. Two-Dimensional Array
3. Multi-Dimensional Array

36. What is the use of ‘\0’ character?
When declaring character arrays (strings), ‘\0’ (NULL) character is automatically added at end. The ‘\0’ character acts as an end of character array.

37. What is ‘C’ functions? Why they are used?
A function is a self-contained block (or) a sub-program of one or more statements that performs a special task when called. To perform a task repetitively then it is not necessary to re-write the particular block of the program again and again. The function defined can be used for any number of times to perform the task.

38. Differentiate library functions and User-defined functions.

<table>
<thead>
<tr>
<th>Library Functions</th>
<th>User-defined Functions</th>
</tr>
</thead>
</table>

www.studentsfocus.com
Sri Vidyag College Engineering & Technology - Virudhnagar

| a) Library functions are pre-defined set of functions that are defined in C libraries. | a) The User-defined functions are the functions defined by the user according to his/her requirement. |
| b) User can only use the function but cannot change | b) User can use this type of function. User can also modify this function. |

39. What are the steps in writing a function in a program.
   a) Function Declaration (Prototype declaration):
      Every user-defined functions has to be declared before the main
   b) Function Callings:
      The user-defined functions can be called inside any functions like main(), user-defined function, etc.
   c) Function Definition:
      The function definition block is used to define the user-defined functions with statements.

40. Give the syntax for using user-defined functions in a program.
    Syntax for using user-defined functions in a program
    Syntax:
    ```
    function declaration;
    function
    main()
    {
    ======  
    ========
    function calling;  (or) function calling;
    ======  
    ========
    }
    function definition;
    ```

41. Classify the functions based on arguments and return values.
    Depending on the arguments and return values, functions are classified into four types.
    a) Function without arguments and return values.
    b) Function with arguments but without return values. c) Function without arguments but with return values. d) Function with arguments and return values.
Define pre-processor in C.
The C Preprocessor is not part of the compiler, but is a separate step in the compilation process. In simplistic terms, a C Preprocessor is just a text substitution tool. We'll refer to the C Preprocessor as the CPP.

Example:
#define Substitutes a preprocessor macro
#include Inserts a particular header from another file

Define Macro in C.
A macro definition is independent of block structure, and is in effect from the #define directive that defines it until either a corresponding #undef directive or the end of the compilation unit is encountered. Its format is: #define identifier replacement

Example:
#define TABLE_SIZE 100
int table1[TABLE_SIZE];
int table2[TABLE_SIZE];

What are conditional Inclusions in Preprocessor Directive?
Conditional inclusions (#ifdef, #ifndef, #if, #endif, #else and #elif)
These directives allow including or discarding part of the code of a program if a certain condition is met. #ifdef allows a section of a program to be compiled only if the macro that is specified as the parameter has been defined, no matter which its value is.

For example:
#define TABLE_SIZE
int table[TABLE_SIZE];
#endif

What you meant by Source file Inclusion in Preprocessor directive?
Source file inclusion (#include)
This directive has also been used assiduously in other sections of this tutorial. When the preprocessor finds an #include directive it replaces it by the entire content of the specified file. There are two ways to specify a file to be included:

1. `#include "file"`
2. `#include <file>`

**PART – B**

1. Explain about the various decision making statements in ‘C’ language.
2. Explain the control statements in c.
3. What are functions? Explain the types of functions in detail with an example program for each type.
4. Define arrays. Explain the array types with an example program for each type
5. What are pointers? When and why they are used? Explain in detail with sample programs.
6. Describe in detail about the Preprocessors in C.
7. Explain function pointers with example.
8. Detailly explain about function with variable number of arguments?