KSU CET

S1 & S2 Notes

2019 Scheme

Manifest the leadership in you

01EST102052002 B

F

Pages: 2

Reg No.: [redacted] Name: [redacted]

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Second Semester B.Tech Degree Examination July 2021 (2019 scheme)

Course Code: EST102
Course Name: PROGRAMMING IN C
(AN Session)

Max. Marks: 100 Duration: 3 Hours

PART A

Answer all Questions. Each question carries 3 Marks Marks

1	Differentiate between system software and application software.	(3)
2	Write an algorithm to find the largest of three numbers	(3)
3	What is the difference between assignment and equality operators?	(3)
4	What is a static variable? When should it be used?	(3)
5	Write a C program to find length of a string without using string handling functions.	(3)
6	What is an array? Illustrate using an example, how a single dimensional array is initialised.	(3)
7	Differentiate between structure and union using an example.	(3)
8	Illustrate the purpose of return statement using an example.	(3)
9	Differentiate between char name[] and char *name in C.	(3)
10	Explain the use of fseek() function.	(3)

PART B

Answer any one Question from each module. Each question carries 14 Marks

11	a)	Draw a flowchart to find the factorial of a number.	(6)
	b)	With the help of a neat diagram explain the functional units of a computer.	(8)

OR

12	a)	List five important registers in CPU. Also state the purpose of each register.	(6)
	b)	Write algorithm and draw flowchart to perform swapping of two numbers.	(8)
13	a)	Explain arithmetic, logical and bitwise operators with examples.	(7)
	b)	Write a C Program to check if a given number is a strong number or not. A strong number is a number in which the *sum of the factorial of the digits is equal to the number itself.* Eg:- 145=1!+4!+5!=1+24+120=145	(7)

OR

b) What do you mean by Formatted Input? Explain in detail the prototype of 'scanf()' function in C including its argument list and return type. (7)

15 a) Explain any 4 string handling functions in C programming. (7)

b) Write a C program to perform linear search on an array of numbers (7)

OR

16 a) Write a C program to find second largest element in an array. (7)

b) Write a C program to check whether a string is palindrome or not without using string handling functions. (7)

17 a) What are different storage classes in C? Give examples for each. (7)

b) Write a C program to: (7)

(i) Create a structure with fields: Name, Address, Date of birth.

(ii) Read the above details for five students from user and display the details

OR

18 a) What is recursion? Write a C program to display Fibonacci series using recursive function. (7)

b) Write a C program to sort N numbers using functions. (7)

19 a) Explain any 5 file handling functions in C. (7)

b) Write a C program to reverse a string using pointers. (7)

OR

20 a) Differentiate between array of pointers and pointer to an array. (7)

b) Write a C program to count number of lines in a text file. (7)

****

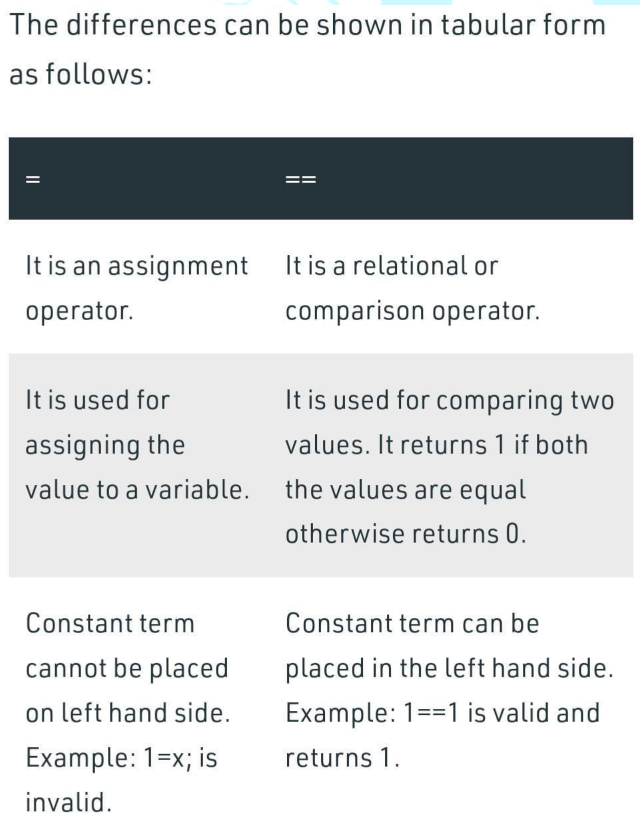
1. System Software is the type of software that is the interface between application software and system. Low-level languages are used to write the system software. System Software maintains the system resources and gives the path for application software to run. An important thing is that without system software, the system can not run. It is a general-purpose software.

Application Software is the type of software that runs as per user request. It runs on the platform which is provided by system software. High-level languages are used to write the application software. It's a specific purpose software. The main difference between System Software and Application Software is that without system software, the system can not run on the other hand without application software, the Low-level maintains system always runs.

2. Algorithm to find greatest number of three given numbers
Ask the user to enter three integer values.
Read the three integer values in num1, num2, and num3 (integer variables).
Check if num1 is greater than num2.
If true, then check if num1 is greater than num3.
If true, then print 'num1' as the greatest number.
If false, then print 'num3' as the greatest number.
If false, then check if num2 is greater than num3.
If true, then print 'num2' as the greatest number.
If false, then print 'num3' as the greatest number.

3. The "=" is an assignment operator is used to assign the value on the right to the variable on the left.

The '==' operator checks whether the two given operands are equal or not. If so, it returns true. Otherwise it returns false.


The differences can be shown in tabular form as follows:

=	==
It is an assignment operator.	It is a relational or comparison operator.
It is used for assigning the value to a variable.	It is used for comparing two values. It returns 1 if both the values are equal otherwise returns 0.
Constant term cannot be placed on left hand side. Example: 1=x; is invalid.	Constant term can be placed in the left hand side. Example: 1==1 is valid and returns 1.

4. static variable is a variable that has been allocated "statically", meaning that its lifetime (or "extent") is the entire run of the program.

static variable is used for a common value which is shared by all the methods and its scope is till the lifetime of whole program. In the C programming language, static is used with global variables and functions to set their scope to the containing file.

5.
```
#include <stdio.h>

void main()
{
   char string[50];
   int i, length = 0;

   printf("Enter a string \n");
   gets(string);

   for (i = 0; string[i] != '\0'; i++)
   {
      length++;
   }
   printf("The length of a string is the number of characters in it \n");
   printf("So, the length of %s = %d\n", string, length);
}
```

6. An array is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier

There are four different ways to initialize one-dimensional array in c programming.

1. Initialize array at the time of declaration
One way is to initialize one-dimentional array is to initialize it at the time of declaration. You can use this syntax to declare an array at the time of initialization.

int a[5] = {10, 20, 30, 40, 50};
a[0] is initialized with 10, a[1] is initialized with 20 and so on.

2. Initialize all elements of an array with 0 (zero)
C program does not assign any default value to declared variables. Same is true for the elements of an array. When we declare an array, all of its elements get initialized with garbage value. If you don't want to get all elements initialized with garbage, you can initialize them with value 0 (zero). You can do this with the help of this syntax.

int a[5] = {0};
With the following syntax, a[0] to a[5] all are initialized with value 0 (zero).

Important Point: You can initialize an array with 0 (zero) only.
If you want to initialize elements with value 10, this syntax will not work.

int a[5] = {10}; //This will not initialize all statements with 10

3. Initialize to define the size of an array

Till now, we declare an array with size. Although, it is also possible to define the size of an array by initializing elements of the array. You can use this syntax for this.

int a[] = {10, 20, 30, 40, 50};

Remember: if you are not defining size inside [ ] brackets, you must initialize the array (it will define array's size).

If you will not define size of an array inside [ ] brackets, and you are also not initializing the array while declaring. Compiler will show an error "size of array is unknown or zero" and program will not compile.

int a[]; //This will cause an error.

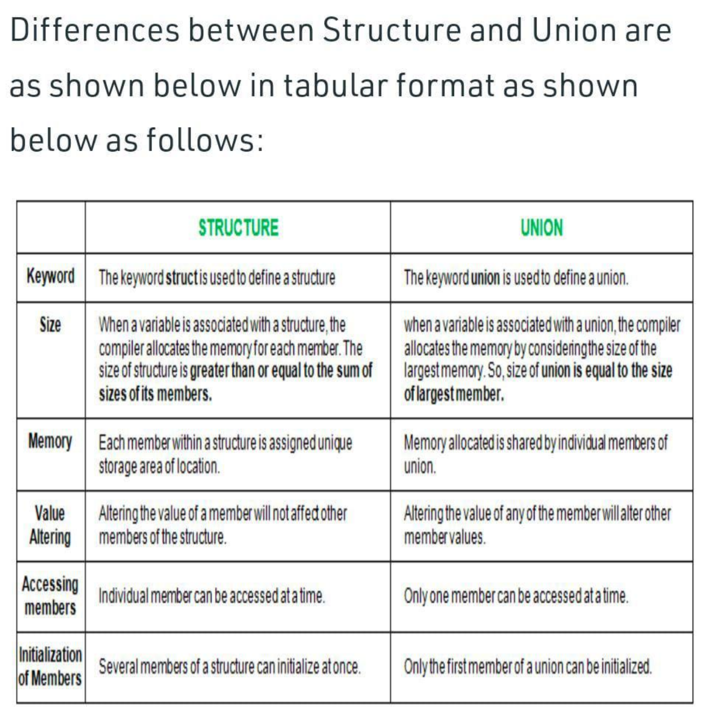
4. Initialize array elements individually

Array is a group of variables, each variable is called element of the array. You can initialize all elements separately as you initialize any other variable. See following syntax.

int a[5];
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[3] = 40;
a[4] = 50;

This is the most useful approach to initialize one-dimensional array in c programming. Most of the time, array elements are initialized with this method as per the requirement of the program.

7.



Differences between Structure and Union are as shown below in tabular format as shown below as follows:

	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by consideringthe size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

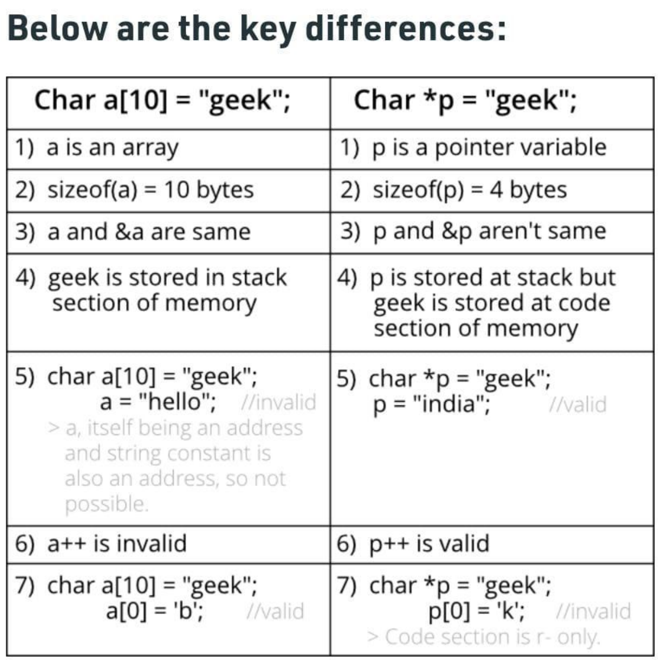
8. The return statement terminates the execution of a function and returns control to the calling function. Execution resumes in the calling function at the point immediately following the call. A return statement can also return a value to the calling function. A return statement causes your function to exit and hand back a value to its caller. The point of functions, in general, is to take in inputs and return something. The return statement is used when a function is ready to return a value to its caller.

Example:

```
void main()
{
   int sum = sumDigits();
   printf("%d\n", sum);
   return;
}

int sumDigits()
{
   int sum = 0;
   int digit;
   for(digit = 0; digit <= 9; ++digit)
   {
      sum += digit;
   }
   return sum;
}
```

9.



Below are the key differences:

Char a[10] = "geek";	Char *p = "geek";
1) a is an array	1) p is a pointer variable
2) sizeof(a) = 10 bytes	2) sizeof(p) = 4 bytes
3) a and &a are same	3) p and &p aren't same
4) geek is stored in stack section of memory	4) p is stored at stack but geek is stored at code section of memory
5) char a[10] = "geek"; a = "hello"; //invalid > a, itself being an address and string constant is also an address, so not possible.	5) char *p = "geek"; p = "india"; //valid
6) a++ is invalid	6) p++ is valid
7) char a[10] = "geek"; a[0] = 'b'; //valid	7) char *p = "geek"; p[0] = 'k'; //invalid > Code section is r- only.

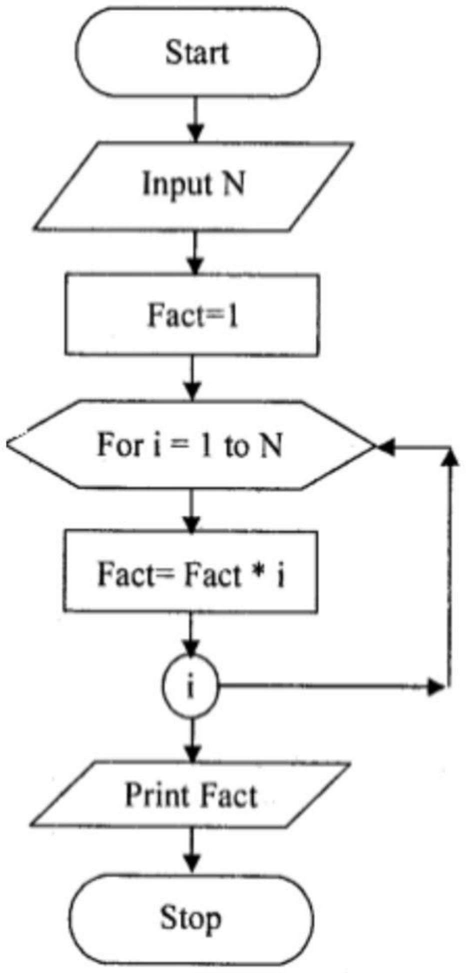
10. fseek() is used to move file pointer associated with a given file to a specific position.
Syntax:

int fseek(FILE *pointer, long int offset, int position)
pointer: pointer to a FILE object that identifies the stream.
offset: number of bytes to offset from position
position: position from where offset is added.

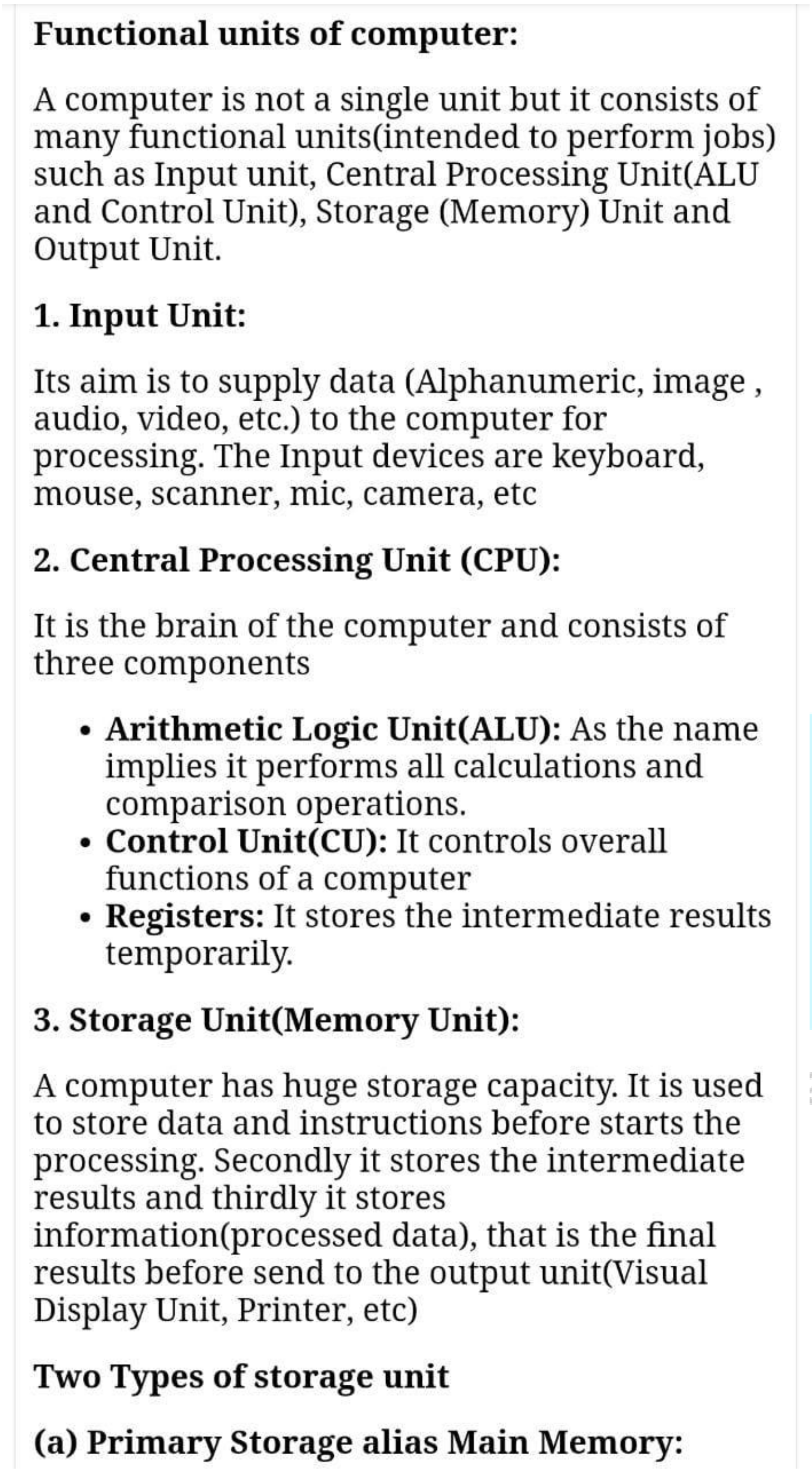
returns:
zero if successful, or else it returns a non-zero value
position defines the point with respect to which the file pointer needs to be moved. It has three values:
SEEK_END : It denotes end of the file.
SEEK_SET : It denotes starting of the file.
SEEK_CUR : It denotes file pointer's current position.

11.
(a)

Flowchart of factorial of a number.



(b)



Functional units of computer:

A computer is not a single unit but it consists of many functional units(intended to perform jobs) such as Input unit, Central Processing Unit(ALU and Control Unit), Storage (Memory) Unit and Output Unit.

1. Input Unit:

Its aim is to supply data (Alphanumeric, image , audio, video, etc.) to the computer for processing. The Input devices are keyboard, mouse, scanner, mic, camera, etc

2. Central Processing Unit (CPU):

It is the brain of the computer and consists of three components

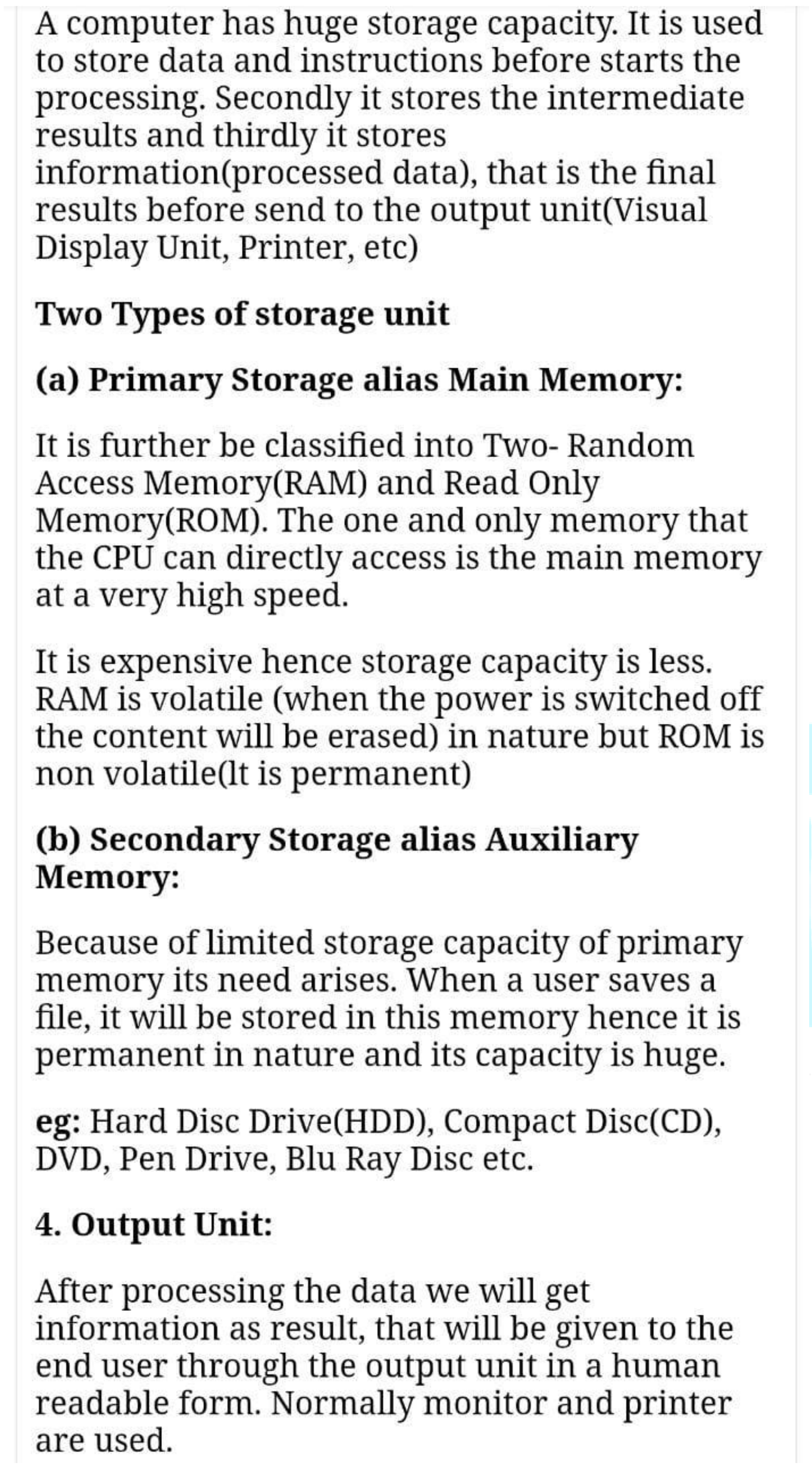
- **Arithmetic Logic Unit(ALU):** As the name implies it performs all calculations and comparison operations.
- **Control Unit(CU):** It controls overall functions of a computer
- **Registers:** It stores the intermediate results temporarily.

3. Storage Unit(Memory Unit):

A computer has huge storage capacity. It is used to store data and instructions before starts the processing. Secondly it stores the intermediate results and thirdly it stores information(processed data), that is the final results before send to the output unit(Visual Display Unit, Printer, etc)

Two Types of storage unit

(a) Primary Storage alias Main Memory:



A computer has huge storage capacity. It is used to store data and instructions before starts the processing. Secondly it stores the intermediate results and thirdly it stores information(processed data), that is the final results before send to the output unit(Visual Display Unit, Printer, etc)

Two Types of storage unit

(a) Primary Storage alias Main Memory:

It is further be classified into Two- Random Access Memory(RAM) and Read Only Memory(ROM). The one and only memory that the CPU can directly access is the main memory at a very high speed.

It is expensive hence storage capacity is less. RAM is volatile (when the power is switched off the content will be erased) in nature but ROM is non volatile(lt is permanent)

(b) Secondary Storage alias Auxiliary Memory:

Because of limited storage capacity of primary memory its need arises. When a user saves a file, it will be stored in this memory hence it is permanent in nature and its capacity is huge.

eg: Hard Disc Drive(HDD), Compact Disc(CD), DVD, Pen Drive, Blu Ray Disc etc.

4. Output Unit:

After processing the data we will get information as result, that will be given to the end user through the output unit in a human readable form. Normally monitor and printer are used.

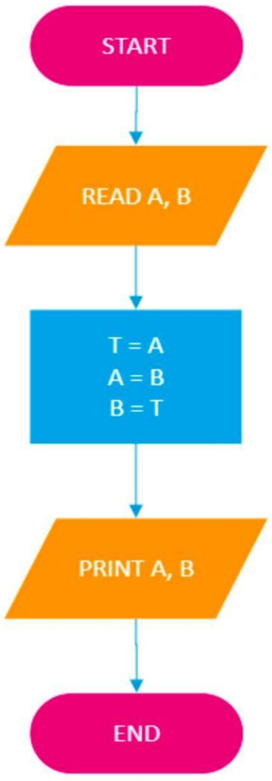
12.
(a) The Data Register (DR) contains 16 bits which hold the operand read from the memory location.
The Memory Address Register (MAR) contains 12 bits which hold the address for the memory location.

The Accumulator (AC) register is a general purpose processing register.

The instruction read from memory is placed in the Instruction register (IR).
The Temporary Register (TR) is used for holding the temporary data during the processing.
The Input Registers (IR) holds the input characters given by the user.
The Output Registers (OR) holds the output after processing the input data.

(b)

Flowchart



```
#include <stdio.h>

int main()
{
   int a = 5;
   int b = 10;
   int t;
   t = a;
   a = b;
   b = t;
   printf("\n a = %d", a);
   printf("\n b = %d", b);
   return 0;
}
```

13.

(a) Arithmetic Operators

The arithmetic operator allows the program to perform general algebraic operations against numeric values.

There are five basic operators present in the C# programming language.

Addition (symbol “+”): Perform the addition of operands.
Subtraction (symbol “-“): Performs subtraction of operands.
Division (symbol “/”): Performs division of operands.
Multiplication (symbol “*”): Performs multiplication on operands.
Modulus (symbol “%”): Returns reminder after the division of integer.
Example:

```
int a = 10;
int b = 5;
int result;
result = a + b;
result = a - b;
result = a * b;
result = a / b;
result = a % b;
```

Relational Operators

Any relation between the two operands is validated by using relational operators. Relational operators return Boolean values. If the relation between two operands is successfully validated then it will return “true” and if the validation fails then “false” will be returned.

Relational operators are mainly used in decision making or for defining conditions for loops.

Greater than operator: (denoted by “>”): Validates greater than the relation between operands.
Less than operator: (denoted by “<“): Validates less than the relation between operands.
Equals To operator: (denoted by “==”): Validates the equality of two operands.
Greater than or equals to (denoted by “>=”): Validates greater than or equals to the relation between the two operands.
Less than or equals to (denoted by “<=”): Validates less than or equals to the relations between the two operands.
Not equal: (denoted by “!=”): Validates not an equal relationship between the two operands.

```
int a = 10;
int b = 5;
bool validate;
validate = a > b;
Console.WriteLine(validate);
validate = a < b;
Console.WriteLine(validate);
validate = a == b;
```

```
Console.WriteLine(validate);
validate = a >= b;
Console.WriteLine(validate);
validate = a <= b;
Console.WriteLine(validate);
validate = a != b;
Console.WriteLine(validate);
```

Assignment Operators

Assignment operators are used for assigning value to a variable. These are generally used before an arithmetic operator.

(i) Equals to (“=”): It is one of the simplest assignment operators. It assigns the value of one operand to another. i.e. the value of the right side operand to the left side operand.

Example: a = b

(ii) Add Equal to the Assignment Operator: As the name suggests it is a combination of plus “+” and equal to “=”. It is written as “+=” and it adds the operand at the right side to the left operand and stores the final value in the left operand.

Example: a +=b means (a = a+b)

(iii) Subtract Equal Assignment Operator: Similar to the add equals, it subtracts the value of the right operand from the left operand and then assigns the value to the left operand.

Example: a -=b means (a = a-b)

(iv) Division Equal to the Assignment Operator: It divides the value of the right operand with the left operand and then stores the result in the left operand.

Example: a /= b mean (a= a/b)

(v) Multiply Equal to the Assignment Operator: It multiplies the value of the right operand with the left operand and then stores the result in the left operand.

Example: a *= b mean (a= a*b)

(vi) Modulus Equals to the Assignment Operator: It finds the modulus of the left and right operand and stores the value in the left operand.

Logical Operators

Logical operators are used for performing logical operations. Logical operators work with Boolean expressions and return a Boolean value. Logical operators are used with the conditional operators in loops and decision-making statements.

Logical Operators and their usage.

#1) Logical AND Operator
Symbol: “&&”

AND operator returns true when both the values are true. If any of the value is false then it will return false.

For example, A && B will return true if both A and B are true, if any or both of them are false then it will return false.

#2) Logical OR Operator
Symbol: “||”

OR operator returns true if any of the condition/operands is true. It will return false when both of the operands are false.

For example, A || B returns true if the value of either of A or B is true. It will return false if both A and B have false values.

#3) Logical NOT Operator
Symbol: “!”

NOT operator is used to reverse the logical conclusion of any condition. If the condition is true then it will return false and if the condition is false then it will return true.

Example, !(A||B) returns false if “A||B” returns true and will return true if “A||B” returns false.

(b)
```
#include<stdio.h>#include<stdlib.h>
int main(int a, char *b[])
{
int number, i, temp, sum = 0, factorial = 1;
number = atoi(b[1]);
temp = number;
while(number != 0)
{
int rem = number%10;
for(i=2; i<=rem; i++)
{
factorial = factorial * i;
}
sum = sum + factorial;
number = number/10;
factorial = 1;
}
if(temp == sum)
printf("YES");
else
printf("NO");
return 0;
}
```

14.

(a)
```
#include<stdio.h>
#include<stdlib.h>
int main(){
int a[10],n,i;
system ("cls");
printf("Enter the number to convert: ");
scanf("%d",&n);
for(i=0;n>0;i++)
{
a[i]=n%2;
n=n/2;
}
printf("\nBinary of Given Number is=");
for(i=i-1;i>=0;i--)
{
printf("%d",a[i]);
}
return 0;
}
```

(b) Formatted I/O functions are used to take various inputs from the user and display multiple outputs to the user. These types of I/O functions can help to display the output to the user in different formats using the format specifiers. These I/O supports all data types like int, float, char, and many more.

What is scanf() function in C

The function int scanf(const char *format, ...); reads formatted data from stdin(standard input device) and stores them in the variables pointed by the additional arguments. Additional arguments must point to variables of the same type as specified in the format.

Function prototype of scanf

int scanf(const char *format, ...);

format : This is a null terminated string that contains Whitespace character, Non-whitespace character and Format specifiers.

additional arguments : As per the format string, the function may expect a sequence of additional arguments, each containing a pointer to allocated storage where the data read from stdin to be stored.

Return value of scanf

On success, scanf function returns the total number of objects successfully read, it may or may not be same as the expected number of items specified in format string.

15.
(a) 1)strcat()
For the cases when one string has to be appended at the end of another string, this function is being used. Function strcat can append a copy of the source string at the end of the destination string. The strcat() is one of the string operations in c which concatenates two strings, meaning it joins the character strings end-to-end. In the strcat() operation, the destination string's null character will be overwritten by the source string's first character, and the previous null character would now be added at the end of the new destination string which is a result of stcrat() operation.

The user has to pass two arguments that are described below:

i) src

ii) dest

Here at the place of "src" string is specified, while at the place of 'dest' the destination string in which we have to append the source string is specified.

Example

```
#include<string.h>

int main()

{

char src[20]= " before";

char dest[20]= "after ";

strcat(dest, src);

puts(dest);

return 0;

}
```

The output will be: after before

2) Function strlen()
One more function of string header file that can be directly used for the strings is strlen(). You can use the function strlen(), the string function in C, when you have to find out the length of any string. The strlen() string functions in c basically calculate the length of a given string. However, one can also write a program manually to find out the length of any string, but the use of this direct function can save your time and the example is given below:

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

```
int main()

{

int length;

char s[20] = “We are Here”;

length=strlen(s);

printf(“\Length of the string is = %d \n”, length);

return 0;

}
```

```
Length of the string is = 11
```

3) Function strcpy()

If you have to copy the content of one string to another string, then this function is being used. Even the null characters are copied in the process. Syntax of the function is strcpy(dest,source). The function can copy the content of one string to another. One example of the function is given below:

```
#include<string.h>

int main()

{

char src[20]= “ Destination”;

char dest[20]= “”;

printf(“\n source string is = %s”, src);

printf(“\n destination string is = %s”, dest);

strcpy(dest, src);

printf (“\ntarget string after strcpy() = %s”, dest);

return 0;

}
```

4.Function strcmp()

To compare two strings to know whether they are same or not we can use strcmp() function.This string functions in c, compares two strings. While comparing the strings takes two parameters into account namely –

str1
str2
On comparing the return value be determined basis the strings setup as shown below.

The function returns a definite value that may be either 0, >0, or <0. In this function, the two values passed are treated as case sensitive means 'A' and 'a' are treated as different letters. The values returned by the function are used as:

i) 0 is returned when two strings are the same

ii) If str1<str2 then a negative value is returned

iii) If str1>str2 then a positive value is returned

Example:

```
#include<stdio.h>

#include<string.h>

int main()

{

char str1[]="copy";

char str2[]="Trophy";

int I,j,k;

i=strcmp(str1, "copy");

j=strcmp(str1, str2);

k-strcmp(str1, "f");

printf("\n %d %d %d",I,j,k);

return 0;
```

```
(b)  int main()
{
  int array[100], search, c, n;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);

  printf("Enter %d integer(s)\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  printf("Enter a number to search\n");
  scanf("%d", &search);

  for (c = 0; c < n; c++)
  {
    If(array[c]==search
    {
      printf("%d is present at location %d.\n", search, c+1);
      break;
    }
  }
  if (c == n)
    printf("%d isn't present in the array.\n", search);

  return 0;
```

16.

(a)
```
#include <stdio.h>
#include <limits.h>

int main()
{
  int arr[50], i, Size;
  int first, second;

  printf("\n Please Enter the Number of elements in an array  :  ");
  scanf("%d", &Size);

  printf("\n Please Enter %d elements of an Array \n", Size);
  for (i = 0; i < Size; i++)
  {
            scanf("%d", &arr[i]);
    }

  first = second = INT_MIN;

  for (i = 0; i < Size; i++)
  {
            if(arr[i] > first)
            {
```

```
                        second = first;
                        first = arr[i];
                }
                else if(arr[i] > second && arr[i] < first)
                {
                        second = arr[i];
                }
        }
        printf("\n The Largest Number in this Array =  %d", first);
        printf("\n The Second Largest Number in this Array =  %d", second);

        return 0;
}
```

(b) #include<stdio.h>

```
int main()
{
   char string[40];
   int length=0, flag=1,i;

   printf("Enter string:\n");
   gets(string);

   for(i=0;string[i]!='\0';i++)
   {
      length++;
   }

   for(i=0;i< length/2;i++)
   {
      if( string[i] != string[length-1-i] )
      {
         flag=0;
         break;
      }
   }

   if(flag==1)
   {
      printf("PALINDROME");
   }
   else
   {
      printf("NOT PALINDROME");
   }
   return 0;
}
```

17.
(a) There are total four types of standard storage classes.

auto	-It is a default storage class.
extern	-It is a global variable.
static	It is a local variable which is capable of returning a value even when control is transferred to the function call.
register	-It is a variable which is stored inside a Register.

Auto Storage Class in C
The variables defined using auto storage class are called as local variables. Auto stands for automatic storage class. A variable is in auto storage class by default if it is not explicitly specified.

The scope of an auto variable is limited with the particular block only. Once the control goes out of the block, the access is destroyed. This means only the block in which the auto variable is declared can access it.

A keyword auto is used to define an auto storage class. By default, an auto variable contains a garbage value.

Extern Storage Class in C
Extern stands for external storage class. Extern storage class is used when we have global functions or variables which are shared between two or more files.

Keyword extern is used to declaring a global variable or function in another file to provide the reference of variable or function which have been already defined in the original file.

The variables defined using an extern keyword are called as global variables. These variables are accessible throughout the program. Notice that the extern variable cannot be initialized it has already been defined in the original file.

Static Storage Class in C
The static variables are used within function/ file as local static variables. They can also be used as a global variable

Static local variable is a local variable that retains and stores its value between function calls or block and remains visible only to the function or block in which it is defined.
Static global variables are global variables visible only to the file in which it is declared.
Example: static int count = 10;
Keep in mind that static variable has a default initial value zero and is initialized only once in its lifetime.

Register Storage Class in C
You can use the register storage class when you want to store local variables within functions or blocks in CPU registers instead of RAM to have quick access to these variables. For example, “counters” are a good candidate to be stored in the register.

Example: register int age;
The keyword register is used to declare a register storage class. The variables declared using register storage class has lifespan throughout the program.

It is similar to the auto storage class. The variable is limited to the particular block. The only difference is that the variables declared using register storage class are stored inside CPU registers instead of a memory. Register has faster access than that of the main memory.

The variables declared using register storage class has no default value. These variables are often declared at the beginning of a program.

(b)

```
#include <stdio.h>
struct student {
   char firstName[50];
   int roll;
   float marks;
} s[5];

int main() {
   int i;
   printf("Enter information of students:\n");

   // storing information
   for (i = 0; i < 5; ++i) {
      s[i].roll = i + 1;
      printf("\nFor roll number%d,\n", s[i].roll);
      printf("Enter first name: ");
      scanf("%s", s[i].firstName);
      printf("Enter marks: ");
      scanf("%f", &s[i].marks);
   }
   printf("Displaying Information:\n\n");

   // displaying information
   for (i = 0; i < 5; ++i) {
      printf("\nRoll number: %d\n", i + 1);
      printf("First name: ");
      puts(s[i].firstName);
      printf("Marks: %.1f", s[i].marks);
      printf("\n");
   }
   return 0;
}
```

18.
(a) Recursion is a process by which a function calls itself directly or indirectly. The corresponding function is called as recursive function. Using recursive algorithms, certain complex problems can be solved quite easily.

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

```
int Fibonacci(int);

int main()
{
  int n, i = 0, c;

  scanf("%d",&n);

  printf("Fibonacci series\n");

  for ( c = 1 ; c <= n ; c++ )
  {
    printf("%d\n", Fibonacci(i));
    i++;
  }

  return 0;
}

int Fibonacci(int n)
{
  if ( n == 0 )
    return 0;
  else if ( n == 1 )
    return 1;
  else
    return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

(b)
```
#include <stdio.h>

void sort_numbers_ascending(int number[], int count)
{
  int temp, i, j, k;
  for (j = 0; j < count; ++j)
  {
    for (k = j + 1; k < count; ++k)
    {
      if (number[j] > number[k])
      {
        temp = number[j];
        number[j] = number[k];
        number[k] = temp;
      }
    }
  }
  printf("Numbers in ascending order:\n");
  for (i = 0; i < count; ++i)
    printf("%d\n", number[i]);
}
void main()
{
  int i, count, number[20];

  printf("How many numbers you are gonna enter:");
  scanf("%d", &count);
  printf("\nEnter the numbers one by one:");

  for (i = 0; i < count; ++i)
    scanf("%d", &number[i]);

  sort_numbers_ascending(number, count);
}
```

19.
(a) In C, you can make use of some functions which are used for file handling purposes. Below is the list of functions:-

fopen()	-Used for opening a new or existing file.
fprintf()	-Used to write data into a file.
fscanf()	-Used to read data from a file.
fputc()-	Used to write a character into a file.
fgetc()-	Used for reading a character from a file.
fclose()-	Closes a file.
fseek()	-Used to set the file pointer to a given position.
fputw()	-Used to write an integer to a file.
fgetw()	-Used for reading an integer from a file.
ftell()-	Used to return a current position.
rewind()-	Used to set the file pointer at the beginning of a file.

(b)
```
#include <stdio.h>
#include <conio.h>
void main()
{
 char *s;
 int len,i;
 clrscr();
 printf("\nENTER A STRING: ");
 gets(s);
 len=strlen(s);
 printf("\nTHE REVERSE OF THE STRING IS:");
 for(i=len;i>=0;i--)
        printf("%c",*(s+i));
 getch();
```

}

20.
(a) Key Differences Between Array and Pointer

1.An array stores the variables of similar data types and the data types of the variables must match the type of array.

2.Conversely, the pointer variable stores the address of a variable, of a type similar to a type of pointer variable type.

3.We can generate an array of pointers i.e. array whose variables are the pointer variables.

4.On the other hand, we can create a pointer that points to an array.

5. An array size decides the number of variables it can store.

6.As against, a pointer variable can store the address of the only variable.

(b)
```
#include <stdio.h>
int main()
{
   FILE *fptr;
   int lcount = 0;
   char fname[40], chr;
   clrscr();
   printf("Enter file name:");
   scanf("%s", fname);

   fptr = fopen(fname, "r");
   chr = getc(fptr);

   while (chr != EOF)
   {
      if (chr == 'n')
      {
         lcount = lcount + 1;
      }
      chr = getc(fptr);
   }
   fclose(fptr); //close file.
   printf("There are %d lines in %s  in a file\n", lcount, fname);
   getch();
   return 0;
}
```