MODEL SOLUTION

AS -4016

B.Te ch (First Semester) course –A

INTRODUCTION TO COMPUTER PROGRAMMING

ANS 1 :

i. Electrically Erasable Programmable Read Only Memory

ii. Four Generation

iii. c

iv. 8

v. High Level

vi. 8 bytes

vii. a is seven

viii. 1 10 2 9 3 8 4 7 5 6

ix. *

x. Lower bound

ANS 2: characteristic of first and second generation of computer

The main features of First Generation are:

- Vacuum tube technology
- Unreliable
- Supported Machine language only
- Very costly
- Generate lot of heat
- Slow Input/Output device
- Huge size
- Need of A.C.
- Non-portable
- Consumed lot of electricity

Some computers of this generation were:

- ENIAC
- EDVAC

- UNIVAC

The main features of Second Generation are:

- Use of transistors
- Reliable as compared to First generation computers
- Smaller size as compared to First generation computers
- Generate less heat as compared to First generation computers
- Consumed less electricity as compared to First generation computers
- Faster than first generation computers
- Still very costly
- A.C. needed
- Support machine and assembly languages

Some computers of this generation were:

- IBM 1620
- IBM 7094

ANS 3 : difference between analog and digital computer :

	Analog	**Digital**
1	Analog signal is a continuous signal which represents physical measurements.	Digital signals are discrete time signals generated by digital modulation.
2	Uses continuous range of values to represent information	Uses discrete or discontinuous values to represent information
3	accuracy of computation is less	Accuracy of computation is more
4	Less programming effort	More programming effort
5	Used in limited no. of application	Used in unlimited no. of application
6	Carry out computation of physical quantity	Carry out computation of numerical digits
7	Precision is limited	Precision is high

ANS 4

I Virtual Memory : It is an illusion given to user that it has very large memory to execute any program of any size , it is implemented using Demand Paging in which the physical memory is divided into equally-sized pages. The memory addressed by a process is also divided into logical pages of the same size. When a process references a memory address, the memory manager fetches from disk the page that includes the referenced address, and places it in a vacant physical page in the RAM. Subsequent references within that logical page are routed to the physical page. When the process references an address from another logical page, it too is fetched into a vacant physical page and becomes the target of subsequent similar references.

II Secondary Memory : **A**lso known as **auxiliary storage**, **secondary storage**, **secondary memory** or **external memory**, is used to store a large amount of data at lesser cost per byte than primary memory. They are two orders of magnitude less expensive than primary storage. In addition, secondary storage does not lose the data when the device is powered down—it is non-volatile. Another difference from primary storage in that it is not directly accessible by the CPU, they are accessed via the input/output channels. The most common form of auxiliary memory devices used in consumer systems is flash memory, optical discs, and magnetic disks.

ANS 5

Difference between compiler and interpreter

Compiler	Interpreter
Compiler take entire program as input	Interpreter take single instruction as input
Intermediate object code is generated	No intermediate object code is generated
Memory requirement is more	Memory requirement is less
Program need not to compile every time	Program need to compile every time
Error are displayed after entire program is checked	Error are displayed after every instruction
Fast	Slow
Exe: c compiler	Exe : basic

ANS 6

I Operating System : It is an software which acts as an intermediate between user and hardware.

Main objective are :

1 effective utilization of all resources

2 make computer user friendly

Function are :

- **Process Management**— The process management activities handled by the OS are—(1) control access to shared resources like file, memory, I/O and CPU, (2) control execution of applications, (3) create, execute and delete a process (system process or user process), (4) cancel or resume a process (5) schedule a process, and (6) synchronization, communication and deadlock handling for processes.

Functions of OS

- **Memory Management**— The activities of memory management handled by OS are—(1) allocate memory, (2) free memory, (3) re-allocate memory to a program when a used block is freed, and (4) keep track of memory usage.

- **File Management**— The file management tasks include—(1) create and delete both files and directories, (2) provide access to files, (3) allocate space for files, (4) keep back-up of files, and (5) secure files.

- **Device Management**— The device management tasks handled by OS are—(1) open, close and write device drivers, and (2) communicate, control and monitor the device driver.

- **Protection and Security**— OS protects the resources of system. User authentication, file attributes like read, write, encryption, and back-up of data are used by OS to provide basic protection.

User Interface or **Command Interpreter**— Operating system provides an interface between the computer user and the computer hardware. The user interface is a set of commands or a graphical user interface via which the user interacts with the applications and the hardware.

II Assembler :

Programming language processor that translates an assembly language program (the source program) to the machine language program (the object program) executable by a computer.
It is so called because in addition to translating it also assembles the machine language program in main memory of the computer and make it ready for execution.

Assembler is system software supplied by computer manufacturer.

ANS 7

(a)

$(A2B2)_{16} = (41650)_{10}$

$(41650)_{10}=(121262)_{8}$

(b)

$(111101.1110111)_{2}=(75.734)_{8}$

ANS 8

1. START

2. Enter number N

3. count=0, i=1

4. if i <=N

4.1 If N%i== 0

4.1.1 Count=count+1

4.2 i=i+1

4.2 repeat step 4.1 and 4.2 till step 4 fails

5 If count == 2

5.1 display given number is prime otherwise

5.2 display given number is not prime

6 stop

ANS 9 Different symbol of flow chart

Start/End Symbol

The terminator symbol marks the starting or ending point of the system. It usually contains the word "Start" or "End."

Action or Process Symbol

A box can represent a single step ("add two cups of flour"), or and entire sub-process ("make bread") within a larger process.

Decision Symbol

A decision or branching point. Lines representing different decisions emerge from different points of the diamond.

Input/Output Symbol

Represents material or information entering or leaving the system, such as customer order (input) or a product (output).

Connector Symbol

Indicates that the flow continues where a matching symbol (containing the same letter) has been placed.

ANS 10

i. Library files in **C:** These are the files which the compiler uses in order to define the functions which have been used in the program and had been declared inside the header file.Like, printf() has its complete definition ,like how it will work etc. in an I/O library! So, the compiler uses that library to get the machine code for printf. This file are there in machine language and store with .lib extension. Library file is binary file so it is very difficult to read and modify.

ii. characteristic of algorithem :

Characteristics of Algorithms:

While designing an algorithm as a solution to a given problem, we must take care of the following five important characteristics of an algorithm.

Finiteness:An algorithm must terminate after a finite number of steps and further each step must be executable in finite amount of time. In order to establish a sequence of steps as an algorithm, it should be established that it terminates (in finite number of steps) on all allowed inputs.

Definiteness (no ambiguity): Each steps of an algorithm must be precisely defined; the action to be carried out must be rigorously and unambiguously specified for each case.

Inputs: An algorithm has zero or more but only finite, number of inputs.

Output: An algorithm has one or more outputs. The requirement of at least one output is obviously essential, because, otherwise we cannot know the answer/ solution provided by the algorithm. The outputs have specific relation to the inputs, where the relation is defined by the algorithm.

Effectiveness: An algorithm should be effective. This means that each of the operation to be performed in an algorithm must be sufficiently basic that it can, in principle, be done exactly and in a finite length of time, by person using pencil and paper. It may be noted that the 'FINITENESS' condition is a special case of 'EFFECTIVENESS'. If a sequence of steps is not finite, then it cannot be effective also.

ANS 11

The syntax for a **switch** statement in C programming language is as follows:

```
switch(expression){
    case constant-expression  :
       statement(s);
       break; /* optional */
    case constant-expression  :
       statement(s);
       break; /* optional */

    /* you can have any number of case statements */
    default : /* Optional */
       statement(s);
}
```

The following rules apply to a **switch** statement:

- The **expression** used in a **switch** statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.
- When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a **break**. If no **break** appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.

Example :

```
#include <stdio.h>

int main ()
{
   /* local variable definition */
   char grade = 'B';

   switch(grade)
   {
   case 'A' :
```

```
      printf("Excellent!\n" );
      break;
   case 'B' :
   case 'C' :
      printf("Well done\n" );
      break;
   case 'D' :
      printf("You passed\n" );
      break;
   case 'F' :
      printf("Better try again\n" );
      break;
   default :
      printf("Invalid grade\n" );
   }
   printf("Your grade is  %c\n", grade );

   return 0;
}
```

ANS 12 : Program to check palindrome

```
#include <stdio.h>

 Void  main()
 {
int n, reverse=0, rem,temp;

printf("Enter an integer: ");
scanf("%d", &n);

 temp=n;
while(temp!=0)

{ rem=temp%10;
reverse=reverse*10+rem;
temp/=10;
 } /* Checking if number entered by user and it's reverse number is equal. */
 if(reverse==n)

 printf("%d is a palindrome.",n);
else
 printf("%d is not a palindrome.",n);

 return 0;

}
```

ANS 13 : Program to find factorial of given number

```
#include <stdio.h>

int main()
{
 int n, count;
 long int factorial=1;
printf("Enter an integer: ");

scanf("%d",&n);

if ( n< 0)
{
printf("Error!!! Factorial of negative number doesn't exist.");
else
{
for(count=1;count<=n;++count) /* for loop terminates if count>n */
{ factorial*=count; /* factorial=factorial*count */
}
printf("Factorial = %d",factorial);
}
return 0;
 }
```

ANS 14

A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before you can use it to store any variable address. The general form of a pointer variable declaration is:

```
type *var-name;
```

Here, **type** is the pointer's base type; it must be a valid C data type and **var-name** is the name of the pointer variable. The asterisk * you used to declare a pointer is the same asterisk that you use for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. Following are the valid pointer declaration:

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

```
#include <stdio.h>

int main ()
{
```

```
   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer variable*/

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );

   return 0;
}
```

ANS 15 Program to find smallest among three numbers

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

Void main()
{
    int a = 0, b = 0, c = 0;
    int minimum = 0;
    printf("\n Enter the first number: ");
    scanf("%d",&a);
    printf(" Enter the second number: ");
    scanf("%d",&b);
    printf(" Enter the third number: ");
    scanf("%d",&c);
    minimum = Minimum(a,b,c);
    printf("\n\t\t\tThe smallest number is: %d",minimum);
    getch();
    return 0;
}
int Minimum(int a, int b, int c)
{
    int minimum = 0;
    if (a < b) //the first number is smaller than the second one
    {
        if (a < c) //the first number is also smaller than the third
            one
        {
            minimum = a;
        }
        else//the first number is larger than the third one
        {
            minimum = c;
        }
    }
    else//the first number is larger than the second one
    {
        if (b < c) //the second number is smaller than the third one
        {
            minimum = b;
        }
        else//the second number is larger than the third one
```

```
        {
            minimum = c;
        }
    }
    return minimum;
}
```

ANS 16: program to swap two value using c all by reference

```
#include <stdio.h>

void swap(int*, int*);

int main()
{
   int x, y;

   printf("Enter the value of x and y\n");
   scanf("%d%d",&x,&y);

   printf("Before Swapping\nx = %d\ny = %d\n", x, y);

   swap(&x, &y);

   printf("After Swapping\nx = %d\ny = %d\n", x, y);

   return 0;
}

void swap(int *a, int *b)
{
   int temp;

   temp = *b;
   *b   = *a;
   *a   = temp;
}
```