SHAMBHUNATH INSTITUTE OF ENGINEERING & TECHNOLOGY

Subject: Programming for Problem Solving **Subject Code: KCS-101**

B.Tech.: 1st Year **SEMESTER: 1st**

SOLUTION- 1st Sessional Examination (2019-2020)

Branch: Computer Science & Engineering

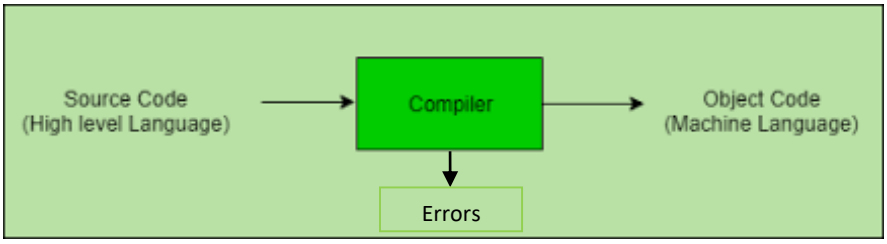
Time – 1 hr. 30 min. **Max Marks – 30**

SECTION – A

Question 1(a): C-Tokens with example:

(i) Keywords: For example, int, float, for, if, etc.
(ii) Identifiers: For example, x, var1, val_2, etc.
(iii) Constants: For example, '1', '2.3', 'c', '{', etc.
(iv) Strings: For example, "Hello_2_c", "Rahul Singh", "Maximum12", etc.
(v) Special Symbols: For example, <, }, $, %, etc.

Question 1(b): Function of Compiler: Compiler is program that converts the code written in high level languages (like C, C++, etc.) into low level language (like assembly level language or machine level language). Machine language code is the binary code which is understandable by the computer machine. Some examples of compilers for C and C++ languages are Portable C, Turbo C/C++, gcc, Visual C++, Intel C++ (ICC), etc.



Question 1(c): Primary data types in C and amount they occupy in memory:

(i) int 2 bytes
(ii) float 4 bytes
(iii) char 1 byte
(iv) double 8 bytes

Question 1(d): Role of clrscr() in a C program: clrscr() is predefined function included in conio.h (console input output header file). When clrscr() function is called in a program everything currently displayed in the console screen such as output of previous programs, output of current program until the invocation of clrscr(), user inputs, error messages, etc. is deleted.

Question 1(e): Output of given program: 0

SECTION – B

Question 2(a): Algorithm: An algorithm is a sequence of finite and well-defined instructions for completing a task or solving a problem. An algorithm is given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state.

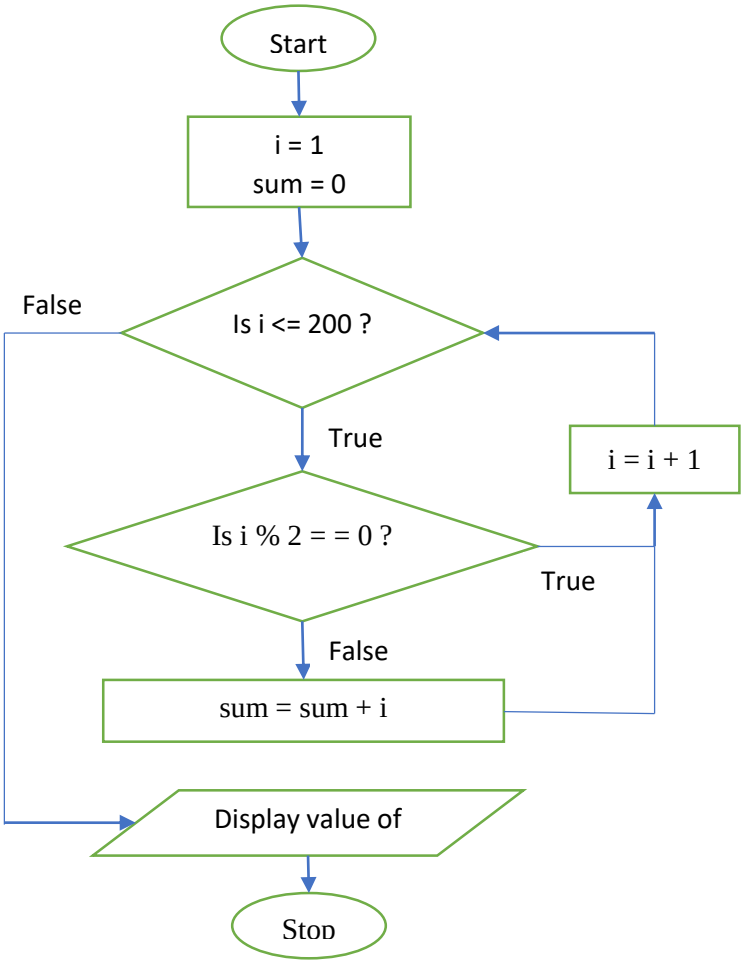
Properties of the algorithm

- **Finiteness.** An algorithm must always terminate after a finite number of steps.
- **Definiteness.** Each step of an algorithm must be precisely defined and must be unambiguous
- **Input.** An algorithm has zero or more inputs
- **Output.** An algorithm has one or more outputs
- **Effectiveness.** An algorithm is also generally expected to be effective. This means that all of the operations of algorithm must be so basic that they can be executed in a finite length of time.

Algorithm to find smallest among three distinct numbers

1. Read three distinct input numbers n1, n2 and n3
2. Take a variable 'small'
3. Compare value of n1 and n2. Whichever is lesser, store it in variable 'small'
4. Now. compare the value of n3 with value of 'small'. Whichever is lesser, store it in variable 'small'
5. Display the value of 'small'.
6. End.

Question 2(b): Flowchart to sum the odd number between 1 to 200.



Question 2(c):

(i). 'break' statement: The 'break' statements in C language causes the termination of enclosing (within which 'break' is used) loop or switch-block immediately. As a break statement is encountered the program control exits from the loop (or switch-block) and jumps to the immediate next statement outside the block.

(ii). ‘continue’ statement: The ‘continue’ statement causes the next iteration of enclosing for, while, or do-while loop to begin. The continue statement is used when we want to skip one or more statements in loop's body and to transfer the control to the next iteration.

Example of ‘break’	**Example of ‘continue’**
# include <stdio.h> # include <conio.h> void main() { int i; for(i=1; i<= 10; i++) { *// when value of i is 5, loop is terminated* if(i = = 5) { break; } printf("%d ", i); } getch(); } **Output:** 1 2 3 4	# include <stdio.h> # include <conio.h> void main() { int i; for(i=1; i<= 10; i++) { *// when value of i is 5, loop begins new cycle* if(i = = 5) { continue; } printf("%d ", i); } getch(); } **Output:** 1 2 3 4 6 7 8 9 10

Question 2(d): (i) Operating System:

- Operating system is a system program that act as an interface between hardware and user.
- It manages system resources.
- It provides a platform on which other application programs are installed.

  Example: Windows, Linux, Unix, DOS, etc.

Functions of Operating System

- Process Management
- Memory Management
- File Management
- I/O Device Management
- Network Management
- Security and Protection

(ii) Storage Devices: Storage device is any hardware capable of holding information either temporarily or permanently.

A. Primary Storgae

Primary memory is the memory which is directly accessed by the CPU during program execution. The programs and data that the CPU requires during execution of a program are stored in primary memory. Primary memory is of two basic types –

- **Volatile memory:** The data in a volatile memory is vanished whenever the power supply to it goes off. RAM is an example of volatile memory
- **Non-volatile memory:** The data in a volatile memory is not vanished whenever the power supply to it goes off. ROM is an example of non-volatile memory

B. Secondary Storage

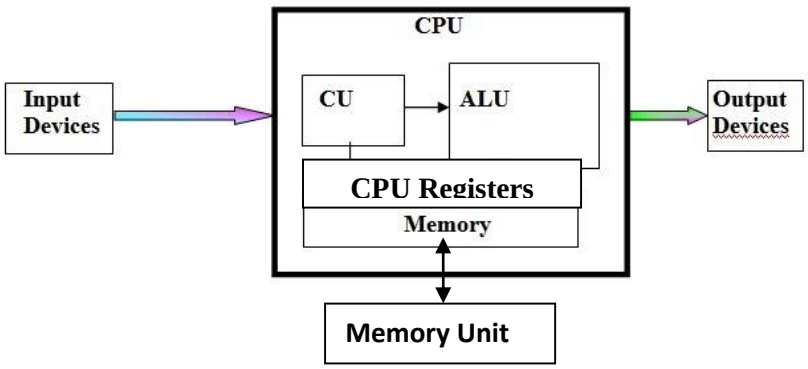
Secondary memory is computer memory that is non-volatile and persistent in nature and is not directly accessed by a computer/processor. Data in secondary memory must be copied into primary storage (also known as RAM) before use. Secondary memories are the slower and cheaper form of memory as compared to primary memory. Secondary memory devices include:

- Magnetic disks like hard drives and floppy disks
- Magnetic tapes
- optical disks such as CDROMs, DVDs, etc.

SECTION – C

Question 3(a): Functional Components of a Digital Computer: Working of a computer includes following three major tasks: Data Input, Processing Input and Data Output. The basic components that helps in performing above mentioned tasks are called as the functional components of a computer. These functional components are:

- Input unit: It takes the input from input devices
- Central processing unit: It does the processing of data
- Output unit: It produces the output and helps in visualizing it.
- Memory unit: It holds the data and instructions during the processing of data.



Block Diagram of a Digital Computer

- **Input Unit:** The input unit consists of input devices that are attached to the computer. These devices take input and convert it into binary language that the computer understands. Some of the common input devices are keyboard, mouse, joystick, scanner, etc.

- **Central Processing Unit (CPU):** The CPU is called the brain of the computer because it is the control center of the computer. CPU processes the information input by the input devices. CPU first fetches the instructions from memory and then interprets them so as to know what is to be done. If required, data is fetched from memory or input device. Thereafter CPU performs the required computation After computation, stores the output or displays it on some output device.

- **Output Unit:** It is composed of output devices attached to the computer. It converts the binary output data coming from CPU to human understandable form. Some examples of different output devices are monitor, printer, plotter, etc.

- **Main Memory Unit:** This memory unit stores data and instructions and also called as Primary Memory, Internal memory, and/or Random Access Memory (RAM). Whenever a program is executed, it's data is copied to this memory and is stored in the memory till the end of the execution.

Question 3(b): <u>Logical Operators:</u> Logical operators allow to join two or more than two test conditions to make a single decision. There are three types of logical operators in C; && (meaning logical AND), || (meaning logical OR) and ! (meaning logical NOT).

Operator	Description	Example	Output
&&	It performs logical conjunction of two expressions. • If both expressions evaluate to True, the overall result is True. • If any of expressions evaluates to False, the overall result is False	int a=0, b=0, c=1, d=2; int e, f, g, h; e = a && b; f = a && c; g = c && a; h = c && d; printf("e=%d\n", e); printf("f=%d\n", f); printf("g=%d\n", g); printf("h=%d", h);	 0 0 0 1
||	It performs a logical disjunction on two expressions. • If either or both expressions evaluate to True, the overall result is True • If both expressions evaluate to False, the overall result is False	int a=0, b=0, c=1, d=2; int e, f, g, h; e = a || b; f = a || c; g = c || a; h = c || d; printf("e=%d\n", e); printf("f=%d\n", f); printf("g=%d\n", g); printf("h=%d", h);	 0 1 1 1
!	It performs logical negation on an expression. • If expression evaluates to True, ! gives False and vice-versa.	int a=0, b=1, c=2; int e, f, g, h; e = !a; f = !b; g = !c; printf("e=%d\n", e); printf("f=%d\n", f); printf("g=%d", g);	 1 0 0

Question 4(a):

(i) Random Access Memory (RAM) –

- It is also called as *read write memory*.
- The programs and data that the CPU requires during execution of a program are stored in this memory.
- It is a volatile memory as the data loses when the power is turned off.

Read Only Memory (ROM) –

- Stores crucial information essential to operate the system, like the program essential to boot the computer.
- It is a non-volatile memory.
- Always retains its data.

Difference between RAM and ROM

Random Access Memory	Read Only Memory
Temporary Storage	Permanent Storage
Volatile Memory	Non Volatile Memory
Writing data is faster	Writing data is slower
Used in normal operations	Used for startup process of computer

(ii) Linker: The Assembler/Compiler generates the object code of a source program and hands it over to the linker. The linker takes this object code and generates the executable code for the program, and hand it over to the Loader. Linking is performed at the last step in compiling a program.

Loader: As the program that has to be executed must reside in the main memory of the computer, it is the responsibility of the loader to load the executable file of a program (generated by the linker) to the main memory for execution. It allocates the memory space to the executable module in main memory.

Difference Linker and Loader

Basis for Comparison	Linker	Loader
Basic	It generates the executable module of a source program.	It loads the executable module to the main memory.
Input	It takes as input, the object code generated by an assembler.	It takes executable module generated by a linker.
Function	It combines all the object modules of a source code to generate an executable module.	It allocates the addresses to an executable module in main memory for execution.

Question 4(b):

The switch statement allows us to execute one code block among many alternatives. The syntax is as follows:

```
switch (expression)
{
   case label1:    // statements
                   break;
   case label2:    // statements
                   break;
   .
   .
   .
   default:        // default statements
}
```

The expression is evaluated once and compared with the values of each case label.

- If there is a match, the corresponding statements after the matching label are executed.
- If there is no match, the default statements are executed.

/* Program to check vowel or consonant */

```
# include <stdio.h>
# include <conio.h>
void main()
{
   int ch;
   printf(“Enter a character”);
   scanf(“%c”,&ch)
   switch (ch)
   {
       case ‘A’:
       case ‘a’:
       case ‘E’:
       case ‘e’:
       case ‘I’:
       case ‘i’:
       case ‘O’:
       case ‘o’:
       case ‘U’:
       case ‘u’:   printf(“It is a vowel”);
                           break;

       default:    printf(“It is a consonant”);
  }
  getch();
}
```

Question 5(a):

/* Program to print the sum of series */

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

void main()
{
   int i, j, x=2, n;
   float sumn=1, muld=1, term=0;

   printf("Enter number of terms in series");
   scanf(“%d”,&n);

   //Loop for number of terms
   for(i=1;i<=n;i++)
   {
      //Calculation of numerator
      sumn = sumn + x;
```

```
        //Calculation of denominator
        muld = muld * x;

        term = term + sumn/muld;
        x++;
    }
    printf("sum of series = %f", term);
    getch();
}
```

Question 5(b):

/* Program to print the pattern */

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

void main()
{
    int i,j,x;
    //Loop for number of rows
    for(i=1;i<=4;i++)
    {
        x=1;

        //Logic to print increasing values in a row
        for(j=1;j<=i;j++)
        {
            printf("%d",x);
            x++;
        }

        x=x-2;

        //Logic to print decreasing values in a row
        for(j=1;j<=i-1;j++)
        {
            printf("%d",x);
            x--;
        }
        printf("\n");
    }
    getch();
}
```