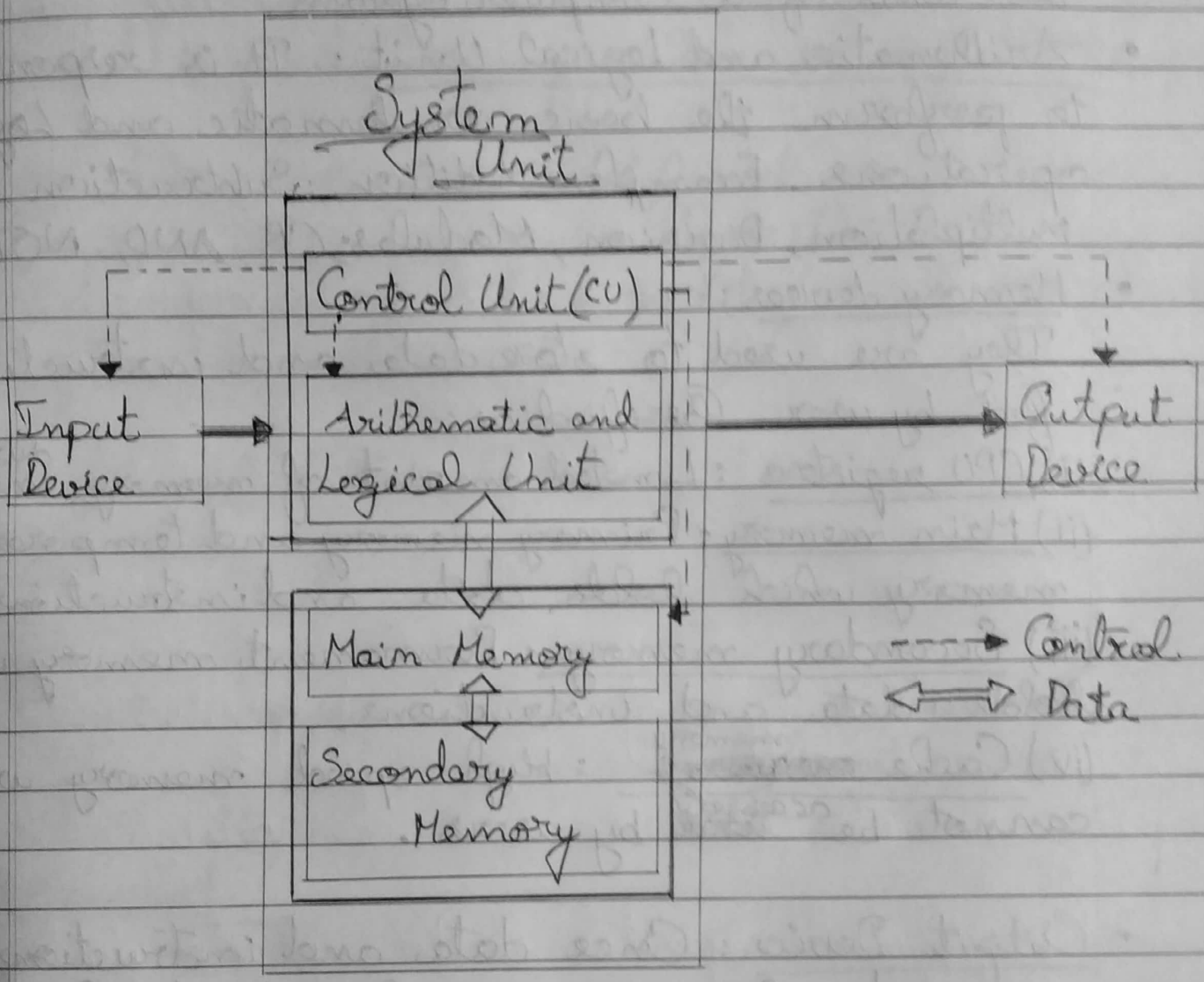
FIRST SEMESTER B.Tech EXAMINATION, DECEMBER-2017

PROBLEM SOLVING THROUGH PROGRAMMING

PART-A

1. a Explain functional block diagram of a computer. 8M



- Input device: This is used to feed data and instructions into the computer. It is connected to System Unit. Examples: Keyboard, mouse, microphone, joystick, digital cameras, etc.

- System Unit: This is responsible for storing and processing of data and instructions.

DATE: PAGE:

The System Unit consists of Central processing Unit (CPU) and Memory devices. The CPU consists of Control Unit (CU) and Arithematic and Logical Unit (ALU).

- Control Unit: It is the important unit in computer. It controls and coordinates the activities of all the units of a computer system.
- Arithematic and Logical Unit: It is responsible to perform the basic arithematic and logical operations. Example: Addition, Subtraction, Multiplation, Division, Modulas; OR, AND, NOT.
- Memory devices:-
  They are used to store data and instructions fed by user. Classified as:-
  (i) CPU registers: Limited amount of memory during execution
  (ii) Main memory: Primary memory and temporary memory which holds data and instructions.
  (iii) Secondary memory: Permanent memory which holds data and instructions.
  (iv) Cache memory: High-speed memory which cannot be accessed by users.

- Output Device: Once data and instructions are processed, the user can choose to display the results on output devices. Example: Monitor, printer, speakers etc.

b Define algorithm and flowchart. Design an algorithm and flowchart for finding area of rectangle.

- An Algorithm can be defined as a step by step procedure to solve a particular problem and is used as a problem solving technique.

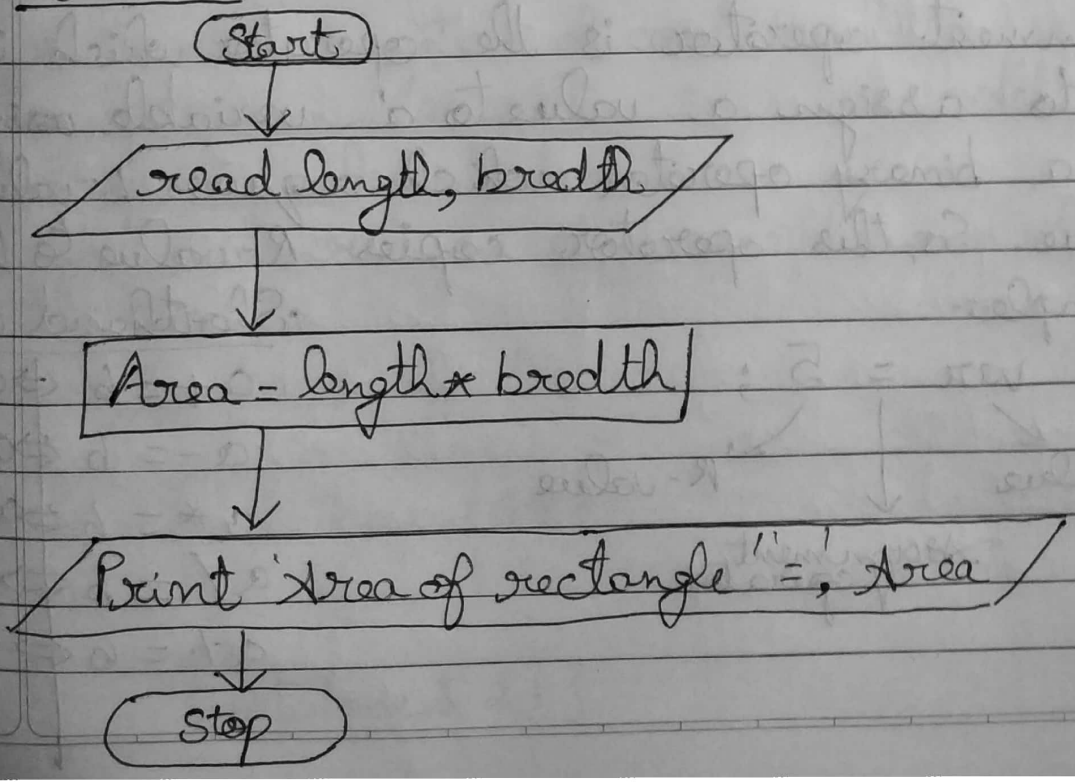
- A Flowchart can be defined as a diagramatic representation of an algorithm. It may also be referred as Blue print / visual representation of an algorithm.

- Area of Rectangle :-

→ Algorithm

Step 1 : Start
Step 2 : Read length, breadth
Step 3 : Compute Area = length x breadth
Step 4 : Print 'Area of rectangle =', Area
Step 5 : Stop

→ Flowchart



DATE PAGE

2. a Define relational operators. Give the output of the following equation.

100/20 <= 10-5 + 100 % 10 - 20 == 5 >= 1 != 20

- Relational operators are the operators which checks the equality of two operands.
Example: == , !=

Output: 100/20 <= 10 -5 +100%10 - 20 == 5 >= 1 != 20
5 <= 5 + 0 - 20 == 5 >= 1 != 20
5 <= -15 == 5 >= 1 != 20
0 == 5 >= 1 != 20
0 >= 1 != 20
0 != 20
1

b. Mention types of assignment operators. Simplify the expression:-
a += b *= c -= 5 when a = 1, b = 3, c = 7

- Assignment operator is the operator which is used to assign a value to a variable in C. It is a binary operator which requires L-value and R-value. So, this operator copies R-value to L-value.
- Example:-

var = 5 ;
var: L-value
=: assignment operator
5: R-value

- Shorthand operator
a += b ⇔ a = a+b
a -= b ⇔ a = a-b
a *= b ⇔ a = a*b
a /= b ⇔ a = a/b
a %= b ⇔ a = a%b

DATE: PAGE

a+=b*=c-=5

=> a = 7

a=1, b=3, c=7

a=a+b a=1+6=7

b=b*c

3*2

c=c-5

7-5

c. Give the syntax for conditional operator. Write a program to find the larger of two number using conditional operator.

- Syntax for conditional operator :-

  expression ? value1 : value2;

- Example :-

```
#include <stdio.h>
void main()
{ int a=4, b=5, result1, result2;
  result1 = a>b ? a:b;
  printf("The result1 = %d", result1);
  result2 = a<b ? a:b;
  printf("The result2 = %d", result2); }
```

Output :-

The result1 = 5 // Greatest

The result2 = 4 // Smallest

3. a. Give the syntax of conditional if-else statement. Write a program to print even or odd number using using if-else statement.

b. Bring out the difference between while loop and do while loop.

a. Syntax of if-else :-

```
if (condition)
{ statement 1;
}
else
{ statement 2; }
```

DATE: PAGE:

- Program to print even or odd number using if-else statement:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int num;
 printf("Enter a number:");
 scanf("%d",&num);
 if(num%2==0)
{ printf("Given number is even!");
}
 else
 { printf("Given number is odd!!");
 }
getch();
}
```

b. Difference between while and do-while

	while	do-while

(i)	Syntax:-
	while(condition)
	{ statement;
	}
		(i) Syntax:-
		do
		{ statement;
		} while(condition);

(ii)	The condition is tested at the beginning of the loop. Therefore, it is called an entry-controlled loop.	(ii) The condition is tested at the end of the loop. Therefore, it is called an exit-controlled loop.

(iii) The while loop may not be executed at all

(iii) The do-while loop is executed at least once.

4. a. Explain with example, the elements of user defined functions.

The elements of user defined functions are :-

(i) Function prototype / declaration
(ii) Function definition
(iii) Function call
(iv) Arguements
(v) Parameters : Formal, Actual.

Example :-

```
#include<stdio.h>
#include<conio.h>
int sum(int, int); // Function prototype or) declaration
void main
{ int a, b, c;
  printf("Enter two numbers :");
  scanf("%d %d", &a, &b);
c=sum(a, b); printf("Sum=", c); // Function call
  getch();
}
int sum(int x, int y)                    // Function definition
{ return x+y;
}
```

actual parameters

Formal parameters

Arguements

(i) Function prototype / declaration is used to declare a function at the beginning

(ii) Function call is used to call the declared function specified to perform a particular task.

DATE: PAGE:

(iii) Function definition is when we define a function which perform the required task.
(iv) Arguements are the variables used in the function parentheses.
(v) (a) Actual parameters are those, which is used at the time of function call.
(b) Formal parameters are those, which is used at the time of function definition.

b. Explain scope rules. Give an example.

- The program part in which a variable or a piece of code can be accessed is known as the scope of the variable/piece of code.
- Local variable :-
  A variable which created/declared within a block or function within a block or function can only be accessed within block/function and the lifetime time of such variable is till the end of the block/function, such variable is called a local variable.
- Global variable : A variable which is declared globally (or) outside of all blocks/functions can be accessed by each and every block/function in the program. The lifetime of such variable is till the end of the program, such variable is called global variable.

Example:-

```
#include <stdio.h>          #include <conio.h>
int a=5;                    int sum(int);
int main() { printf("a=%d", a);
int b=10,c; printf("c=%d", sum(b)); getch(); }
int sum(int b1) { return a+b1; }
```

From the above example it is clear that a is a global variable accessed by both main() & sum() function, whereas b, c are local variables accessed by main() and b is also a local variable accessed by sum() only.

PART - B

5. a. What are pointers in C? How they are declared in C?

A pointer in C is a variable which holds the address of another variable. It can only store the address of another variable. Pointers are used to point to variables such as :-

(i) variable of basic datatype
(ii) array
(iii) functions
(iv) structure
(v) union.

Address of variables is given as: &a

So the declaration of the pointer is given by:-

ptr = &a;

Example :-

```
int a;
int *b;
b = &a;
```

DATE: PAGE:

b. Write a program to return pointer to find the largest of two numbers.

```
#include <stdio.h>
int largest (int *, int *)
void main()
{ int a,b, *c, *d, e;
  printf("Enter two numbers:-\n");
  scanf("%d%d", &a, &b);
  c = &a, d = &b;
e = largest(c, d);
  printf("Largest of %d & %d is %d", a, b, e);
getch();
}
int largest (int *x, int *y)
{
return *c > *d ? *c, *d;
}
```

DATE PAGE

c. a=5, b=7;
p→ q→

(i) ++a = 6
(ii) ++(*p) = 6 // If a=5
++(*p) = 7 // If a=6
(iii) --(*q) = 6
(iv) -b = -7 // If b = 7
-b = -6 // If b=6

6. a. What are two dimensional arrays? Write a function to print matrix of size m x n.
b. Explain in detail command line arguements with example.

• Two dimensional array is a matrix which contains rows(m) and columns(n) in two dimension.

Example: int A[5][10];

function to print matrix of size m x n

```
void array (int m, int n) int A[][]
{ int i,j; int A[m][n];
  Printf (" The given matrix :-\n");
  for(i=0; i<m; i++)
  {for(j=0; j<n; j++)
  {printf ("%d \t", A[i][j]}
   printf ("\n");
  }
}
```

b. It is possible to pass some values from the command line to your C programs when they are executed. These values are called command line arguements, many times they are important.

6.b Example:-

```
#include <stdio.h>
int main (int argc, char *argc[1])
{ if (argc == 2)
{ printf ("The argument supplied is %s\n", argc[1]);}
else if (argc > 2)
{ printf ("Too many arguments supplied.\n");}
else
{ printf ("One argument expected.\n"); } }
```

Output :-

```
$./a.out testing                    // when the above code is
The argument supplied is testing    // executed with single argument
```

DATE: PAGE:

7. a. Define structures in C. Given the structure declaration and initialization.
b. Write a program to create STUDENT details structure having fields name, USN, marks. Read STUDENT records and display the details of student information in ascending order.

Ans:- a. A Structure is a collection of variables of different data types referenced under a common name. Structure is a user derived data type.

Syntax :-

```
struct STUDENT
{
    char name[20];
    int USN;
    float marks;
} st;
```

```
struct (structurename)
{
  (data types);
} (structure variable);
```

It can initialised in many types :- (for the above example)

① Type
st = {"ASWIN", "268", 50.02};

② Type
st.name = "ASWIN"; st.USN = 268; st.marks = 50.02;

③ Type
By getting values from user and initialising it

```
printf (" Enter name: ");
scanf (" %s", & st.name);
printf (" Enter USN: ");
scanf (" %d ", & st.USN);
printf (" Enter marks: ");
scanf (" %f ", & st.marks);
```

b.
```
#include <stdio.h>
struct STUDENT
{
  char name[20];
  int USN;
  float marks;
} st[10];
int main ()
{ int i;
  for (i=0; i<10; i++)
  { printf (" Enter information of student: %d", i+1);
    printf (" \n Enter name: ");
    scanf (" %ds", &st.name);
    printf (" \n Enter USN: ");
    scanf (" %d ", &st.USN);
    printf (" \n Enter marks: ");
    scanf (" %f", & st.marks);
    printf (" \n");
  }
  for (i =0 ; i < 10; i++)
{ printf (" Information of student: %d", i+1);
  printf (" \n Name: %s", st.name);
  printf (" \n USN: %d", st.USN);
  printf (" \n marks: %f", st.marks);
  printf (" \n");
  }
  return 0;
}
```