LAB ASSIGNMENT NO.2

CONTROL STRUCTURES

SHREYAS SAWANT

D2A-55

AIM: Programming using C language

THEORY:

Program Control: Program begins execution at the main() function. Statements within the main() function are then executed from top-down style, line-by-line. However, this order is rarely encountered in real C program. The order of the execution within the main() body may be branched. Changing the order in which statements are executed is called program control. Three types of program control:

1) Sequence control structure

2) Structural control structure for example, if, if-if-else, if-else-if and switch-case-break, goto.

3) Repetition (loop) such as for, while and do-while.

Statements: Expressions, when terminated by a semicolon, become statements. Example, x=5;
One can construct compound statements by putting statements and declarations in Braces { and }. Also called blocks.

Decision Control Statements: It disrupts or alters the sequential execution of statement of program depending on the test condition in program. For example, if, if-if-else, if-else-if and switch-case-break, goto.

Labelled Statements: One can give a label to any statement. The form is 'Identifier: statement.' There are other labelled statements like *case* and *default*, to be used with switch statement.

If-else statement: The if-else statement expresses simplest decision making. The syntax is

```
if (expression)
  statement1
else
  statement2
```

The expression is evaluated and if it is nonzero (true) then the $statement_1$ is executed. Otherwise, if else is present $statement_2$ is executed. Expression can just be numeric and need not be conditional expression only. $Statement_1$ and $statement_2$ may be compound statements or blocks. Each else is associated with closest previous else-less if.

Switch case: The syntax of switch statement is

```
switch (expression) {
case const-expression1 : statement1
case const-expression2 : statement2
 :
 :

default : statementn
```

All case expressions must be different. Expression must evaluate to an integer. First the expression is evaluated. Then the value of expression is compared with the case expressions. The execution begins at the case statement, whose case expression matches. All the statements below are executed. If default is present and if no other case matches then default statement is executed.

Loops: Repetition of statements in known as loop. 'C' programming language provides us with three types of loop constructs:

1.The while loop

2. The do-while loop

3. The for loop

1. While Loop: It is entry-controlled loop i.e., first the condition is evaluated before processing to the body of loop. If the condition is true then control moves to body of the loop. The body of loop is executed till the condition evaluates becomes false. Once the condition evaluates to false the control goes outside of loop and statements after the loop are executed.

Syntax: *while (condition)*

```
{
  Statements ;
}
```

2. Do-while Loop: It is exit-controlled loop i.e., the loop runs once before the condition is evaluated. The control goes out of the loop when condition is evaluated to be false and subsequent statements are executed.

Syntax: *do*

```
{
  Statements ;
} while (condition);
```

3. For Loop: The initial value of the for loop is performed only once. The condition is a Boolean expression that tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned. The incrementation/decrementation increases (or decreases) the counter by a set value.
Syntax: *for (initial value; condition; incrementation or decrementation)*

```
{
  Statements ;
}
```

Break: It provides an early exit from the loops and switch statement.

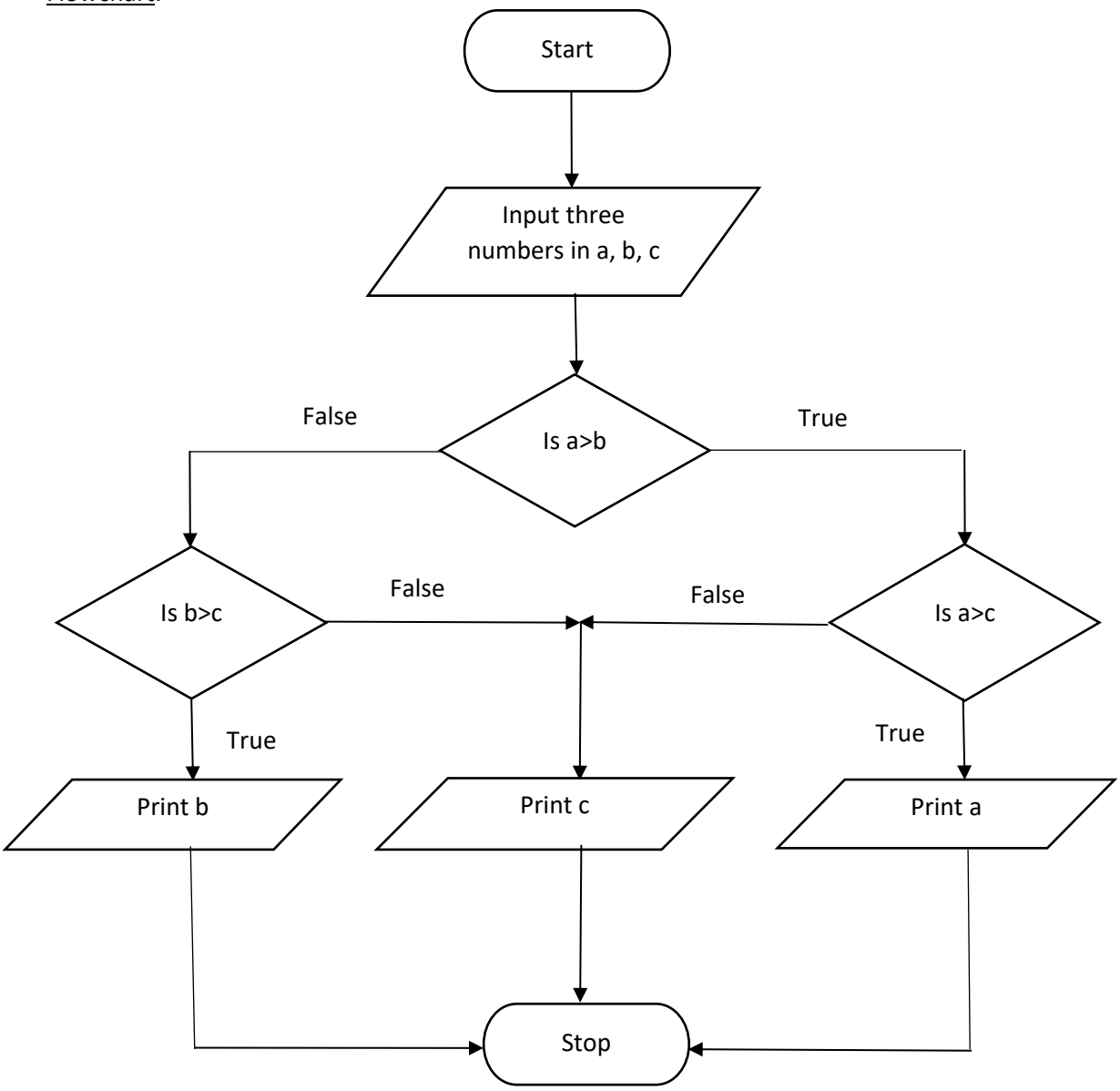
Continue: It causes the jump to end of loop body skipping statements only once. The loop may continue.

SELECTION:
a) Write a program to accept three numbers and display the greatest number.

Variable List:
a= random number
b= random number
c= random number

Algorithm:
Step 1: Start
Step 2: Input values of a, b, c.
Step 3: Check if a is greater than b
    If true, then check if a is greater than c.
        If true, print 'a' as the greatest number.
        If false, print 'c' as the greatest number.
    If false, then check if b is greater than c.
        If true, print 'b' as the greatest number.
        If false, print 'c' as the greatest number.
Step 4: Stop

Flowchart:



Code:

```
#include <stdio.h>

int main()
{   int a,b,c;
    printf("Enter three numbers ");
    scanf("%d %d %d",&a,&b,&c);
    if (a>=b)
    { if(a>=c)
        printf("%d is the greatest among the three",a);
        else
            printf("%d is the greatest among the three",c);
    }
    else
    { if(b>=c)
        printf("%d is the greatest among the three",b);
        else
            printf("%d is the greatest among the three",c);
    }

    return 0;
}
```

Output:

```
Enter three numbers 45 67 8
67 is the greatest among the three
```

CONDITIONAL OPERATOR:

a) Write a program to check whether the character entered through the keyboard is a lowercase alphabet or not.

Variable List:

ch = alphabet to check whether it is uppercase or lowercase.

Algorithm:

Step 1: Start

Step 2: Input an alphabet in ch.

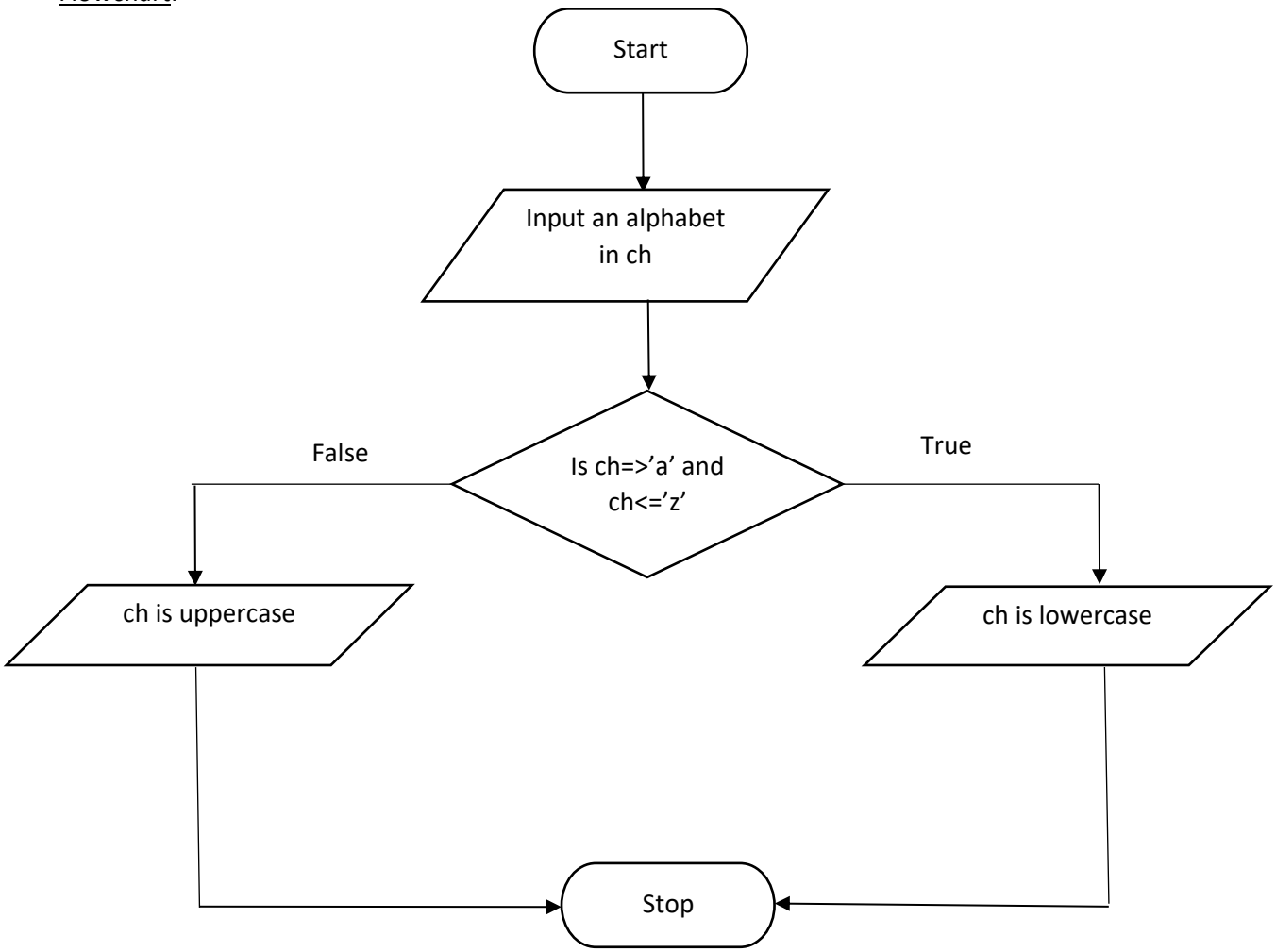
Step 3: Check if ch lies between lowercase alphabet ‘a’ and ‘z’.

If true, then print ‘ch’ is lowercase.

If false, print 'ch' is uppercase.

Step 4: Stop

Flowchart:



Code:

```
#include <stdio.h>

int main()
{   char ch;
    printf("Enter an alphabet ");
    scanf("%c",&ch);
    (ch>='a'&& ch<='z')?printf("%c is lowercase",ch):printf("%c is uppercase",ch);
}
```

Output:

```
Enter an alphabet f
f is lowercase
```

SWITCH CASE

b) Write a program to scan a character from keyboard, if character is ‘a’ or ‘A’ print ‘Apple’ if character is ‘b’ or ‘B’ print ‘Bat’, if character is ‘c’ or ‘C’ print ‘Cat’

Variable list:

ch= to store input character

Algorithm:

Step 1- Start

Step 2- Accept character in ch

Step 3- Use ch expression in switch statement

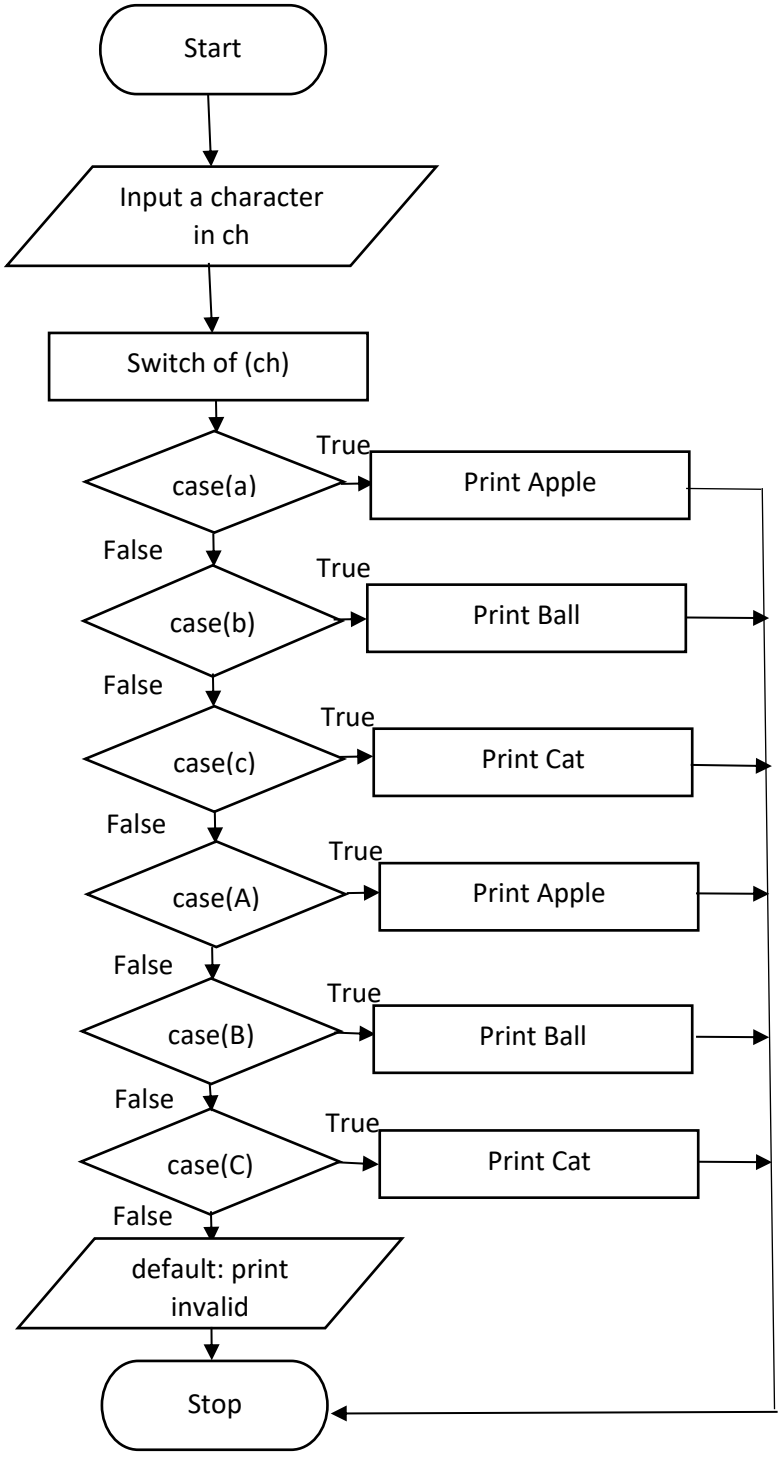
If ch=’a’or ch=’A’ print Apple

If ch=’b’or ch=’B’ print Bat

If ch=’c’or ch=’C’ print Cat

Step 4- Stop

Flowchart:



Code:

```
#include <stdio.h>

int main()
{   char ch;
    printf("Enter a character ");
    scanf("%c",&ch);

    switch(ch)
   {case 'a': printf("Apple ");break;
    case 'b': printf("Bat ");break;
    case 'c': printf("Cat ");break;
    case 'A': printf("Apple ");break;
    case 'B': printf("Bat");break;
    case 'C': printf("Cat ");break;
    default:printf("Invalid");
    }
}
```

Output:

```
Enter a character C
Cat

Enter a character f
Invalid
```

ITERATION

a) Write a program to print all three-digit Armstrong nos.

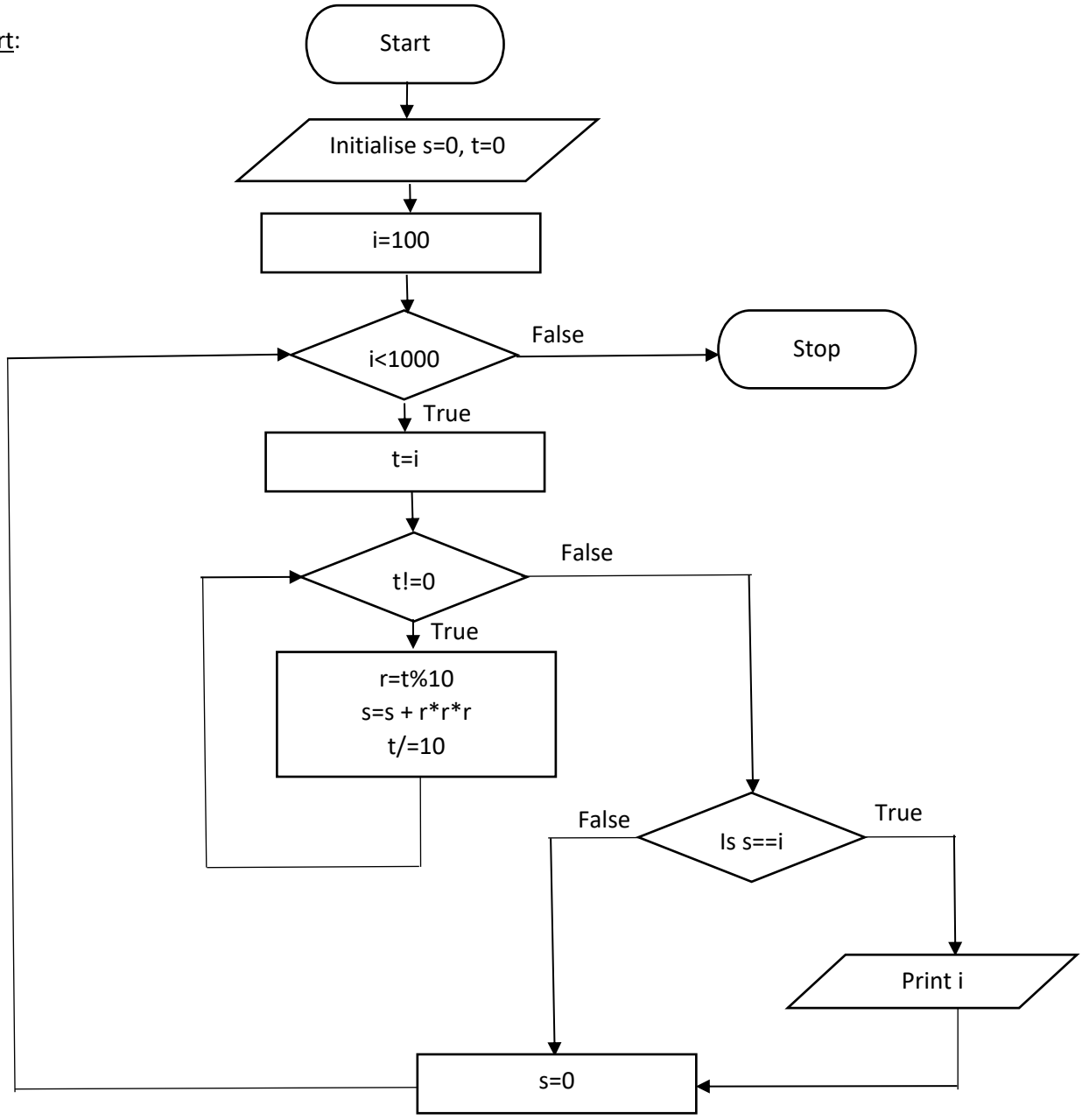
Variable List:

s=to store sum of cube of digits temporarily
t= temporary variable
i= loop counter
r= to store digits of number

Algorithm:

Step 1- Start
Step 2- Initialise s=0,t=0
Step 3- Initialise i=100. Repeat steps 3,4,5,6,7 until i=1000.
Step 4- Equate t to i
Step 5- Repeat until t!=0
    r=t%10
    s=s+ r*r*r
    t=t/10
Step 6- Check whether 's' is equal to the 'i'
    If true then print the i
Step 7- s=0, i=i+1
Step 8- Stop

Flowchart:



Code:

```
#include <stdio.h>

int main()
{   int s=0,t=0;
    printf("The three digit armstrong numbers are: ");

    for(int i=100;i<1000;i++)
    {  t=i;
        while(t!=0)
      {int r=t%10;
       s=s+r*r*r;
       t/=10;
      }
      if(s==i)
      printf("%d ",i);
      s=0;
    }
}
```

Output:

The three digit armstrong numbers are: 153 370 371 407