Note on switch and goto statements

Switch

The switch statement is a multi-way decision statement. It causes a particular group of statements to be chosen from several available groups.

The general form is

```
switch (expression)
{
  case value-1 : statements
  case value-2 : statements
  .....
  default : statements
}
Statement-X;
```

Here **switch** is a keyword. The expression followed by the keyword **switch** is an expression that results in an **integer value** when evaluated. The value-1, value-2, ..., value-n can be either an integer constant or character constant or constant expression.

Ex-1:

```
void main()
{
  int i;
  printf("\nEnter an integer :");
  scanf("%u" , &i);
  if ( i == 1)
    printf("\n Entered 1");
  else if ( i == 2)
    printf("\n Entered 2");
  else if ( i == 3)
    printf("\n Entered 3");
  else
    printf("\n Entered other than 1, 2 or 3",);
}
```

```
void main()
{
  int i;
  printf("\nEnter an integer :");
  scanf("%u" , &i);
  switch(i)
  {
    case 1:
            printf("\n Entered 1");
            break;
    case 2:
            printf("\n Entered 2");
            break;
    case 3:
            printf("\n Entered 3");
            break;
    default:
        printf("\n Entered other than 1, 2 or 3",);
  }
}
```

EX-2 :

```
void main()
{ char c;
  printf("\n Enter a character");
  scanf("%c",&c);
  switch(c)
  {
    case 'r':
    case 'R': printf("\n RED");
              break;
    case 'b':
    case 'B' : printf("\n BLUE");
              break;
    default: printf("\n No match found");
  }
}
```

EX-3 :

```
void main()
{ int n;
  printf("\n Enter a number ");
  scanf("%d",&n);
  switch(n%2)
  {
    case 1: printf("\n ODD");
            break;
    case 0: printf("\n EVEN");

  }
}
```

EX-4 :

```
void main()
{ int n;
  printf("\n Enter a number ");
  scanf("%d",&n);
  switch(n%2)
  {
    case 0+1: printf("\n ODD");
              break;
    case 0: printf("\n EVEN");

  }
}
```


Note on switch and goto statements

Advantage of Switch over if-else:

If we need to choose one among several alternatives, **switch** works faster than an equivalent **if-else** ladder. Thus a **switch** with 10 cases would work faster than an equivalent **if-else** ladder. This is because the compiler generates a **jump table** for a **switch** during compilation and during execution it simply refers to the jump table to decide which case should be executed, rather than actually checking which case is satisfied.

Disadvantages of Switch over if-else:

1. A **float expression** cannot be tested using a switch.
2. Cases can never have **variable expressions** (e.g. it is wrong to write **case a + 3:**).
3. Multiple cases cannot use same expressions. Thus following **switch** is **illegal**.

```
switch (a)
{
  case 3: printf("\n Hello");
          break;
  case 1 +2: printf("\n Hi");
}
```

Prog 1: WACP to find grade of a student who appeared in three subjects, each having full marks 100. (>=80 means grade E, >=60 but <80 means grade A, >=50 but < 60 means grade B else fail).

```
#include <stdio.h>

int main()
{
        int m1,m2,m3,avg;
        printf("\n Enter marks");
        scanf("%d%d%d",&m1,&m2,&m3);
        avg=(m1+m2+m3)/3;
        switch(avg/10)
        {
                case 10:
                case  9:
                case  8: printf("\n Grade E");
                         break;
                case  7:
                case  6: printf("\n Grade A");
                         break;
                case  5: printf("\n Grade B");
                default:  printf("\n Fail");
        }
        return(0);
}
```

Escape sequence: Certain non-printing characters as well as double quote("), single quote('), question mark (?) etc. are represented in terms of escape sequences. These sequences look like two characters but represent only one character.

Example:

1. \n represents newline
2. \t represents horizontal tab
3. printf("\n Press \"Enter\" to continue");

The output of example 3 will be as follows

Press "Enter" to continue

Compiled by Alok Basu for CSE & IT students of SIT

Note on switch and goto statements

goto statement :

It is used to alter the normal sequence of program execution by transferring the control to some other part within the current function.

Example:

```
#include <stdio.h>
int main()
{
  int n,i=1,s=0;
  while(i<=3)
  {
    again: printf("\nEnter no.");
           scanf("%d",&n);
           if(n<0)
             goto errormsg;
           s=s+n;
           i++;
  }
  printf("\n Sum=%d",s);
  goto end;
  errormsg: printf("Negative no. entered");
            goto again;
  end: return(0);
}
```

Advantage of using goto:
It allows us to exit completely from deeply nested loops if an error occurs.

Disadvantages of using goto:

1. Logic becomes complicated.
2. Debugging becomes difficult;
3. Readability gets reduced.
