ID (7832)
Name: Jamal Faraz
Password (6093)
TEACHER
Eng. Syed Ashraf Ali
DATE
29 Sep 2020
ASSIGNMENT
Final term exam.
BE (Civil):-

Question #1

(a) Write a program for yours Grading System using if-else statement:

IF-ELSE Statement:

Some times we need to handle two alternatives in our code.

→ The C++ syntax of the if-else statement is:

```
if ( logical expression)
{
// Block of code to execute if
```

expression is true }
else
{
// Block of code to execute if expression is false }.

> Programming style Suggestions:

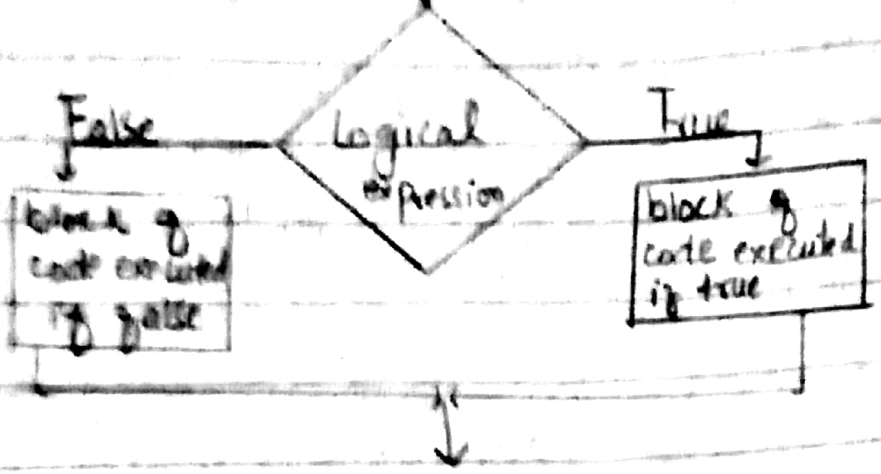
→ Type the "if line" and the "else line" and the {} brackets so they are vertically aligned with each other.

→ Do not put a semi-colon after "else line" or "if line" or you will get very strange run time errors.

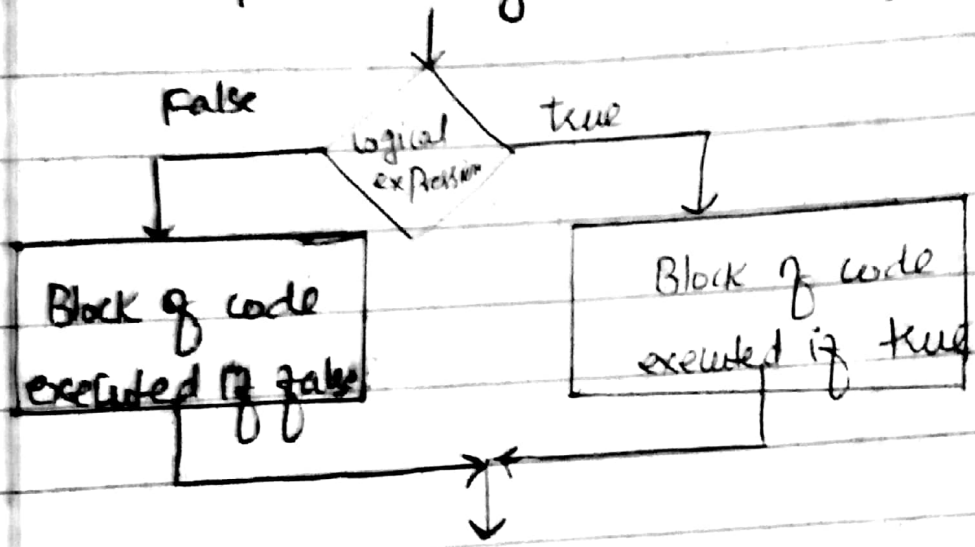
→ The two blocks of code should be intended 3-4 spaces to aid program readability.

→ If either block of code is only one line long the {} brackets can be omitted.

→ We can visualized the program's if else statement decision process using a "flow chart" diagram.



→ If logical expression is true, we take one path through the diagram.



→ If logical expression is false, we take one path through the diagram (executing the other block of code); reverse of the first one.

```
// Simple if-else example
if ((a>0) && (b>0))
{
c= a/b;
a= a-c;
```

```
} else
{
c = a*b;
a = b+c;
}
```

example

```
// ugly if-else
if (a<b){
  c = a*3;
  a = b-c;} else
  a = c+5;
```

This code is nice and short but it is hard to read.

```
// pretty if-else example.
if (a<b)
{
  c = a*3;
  a = b-c;
} else
  a = c+5;
```

This code takes more lines but it is easier to read.

(Part b) Differentiate between if statement and if-else statement.

Difference blw if & if-else

If-Statement

We have two or more if statements inside each other to check multiple conditions.

→ These are called nested if statements.

Use indentation to reflect nesting and aid readability.

Typically indent 3-4 spaces or one tab.

→ Need to take care when matching up {} brackets.

→ This way you can decipher the nesting of conditions.

```
if (logical expression1)
{
    if (logical expression2)
    {
        // statements to execute
    }
}
```

If-else Statement

→ Sometimes we need to handle two alternatives in our code.

The C++ syntax of if-else statement is:

```
if (logical expression)
{
    // block of code to execute if expression is true
}
else
{
    // block of code to execute if expression is false
}
```

→ They are vertically aligned with each other.

→ Do not put semi-colon after 'else line'

→ The two block of code should be intended 3-4 spaces to aid program readability

If- statements

→ If expression 1 true and expression 2
false }
} else
{
// statement of execute if expression
1 false
}.
// Simple nested if example.
cin>> a>> b
if (a<b)
{
cout << "A is smaller than B\n";
if ((a>0) && (b>0))
cout << "A and B both are positive \n";
else
cout << "A or B or both are negative \n";
}
ugly nested if example:
if (a>0) {
if (b<0) {
a = 3*b;
c = a+b; }} else a = 5*a; c = b/a; }
It is hard to see what condition the
else code goes with

Q2) Write a Program to display a menu to perform various functions using "switch statements".

Switch Statements:

The switch statement is convenient for handling multiple branches based on the value of one decision variable;

→ The program looks at the value of decision variable.

→ The program jumps directly to matching case label.

→ The statement following the case label executed.

→ The main advantage of switch statement over a sequence of if-else statements is that it is much faster;

program to display a menu to perform various function:

we can use switch statement to handle menu and also to display a menu to perform various funtions. for example:

// Simulate bank deposits and

and withdrawal;

```
#include <iostream.h>
main()
{
  // Local variable
declarations
// Print command prompt
// Read user input
// Handle banking
} command
```

for first version of program we just write comments in the main program to explain our approach.

```
// Local variable
declarations
int command=0;
// Print Command prompt
Cout << "Enter command
number:\n"
;
// Read users input
Cin >> Command;
...
```

For the next version of Program we add the code to read the user command.

```
// handle banking commands

switch (command)
{
case 0: // Quit code
 break;
case 1: // Deposit code
 break;
case 2: // withdraw code
 break
case 3: // Print balance
 Code
 Break;
}
```

when we add the skeleton of the switch statement to handle the user command

```
// simulate bank deposits
and withdrawals.
#include <iostream.h>
main()
{
// local variable declarations
int command = 0;
Int money = 0;
int Balance = 100;
// Print command prompt
cout << "Enter Command number\n"
```

```
<< "0- quit\n"
<< "1- deposit money\n"
<< "2- withdraw money\n"
<< "3- Print balance\n"
```

→ In final version add bank account variables & add code to perform baking operation

```
// Read & handle baking commands;
 Cin >> Command;
Switch (Command)
{
Case 0; // Quit Code
Cout << "see you later!" << endl;
break;
Case 1: // Deposit Code
Cout << "Enter deposit amount:";
Cin >> Money;
Balance = Balance + Money;
break
Case 2; // with draw code
Cout << "Enter withdraw amount;
Cin >> Money;
 Balance = Balance - Money;
 break;
```

```
case 3: // print balance code
cout << " current balance = " << Balance
<< endl;
break;
default: // Handle other values
cout << " oops try again " << endl;
break; }
// print " final balance = " << Balance
<< endl; }.
```

(b) Differentiate b/w "Switch" statement and "Nested if-else statement.

Switch statement:

- It is convenient for handling multiple branches based on the value of one decision variable;
- While in nested if-else statements we need to handle two alternatives in our code.
- The switch statement is based on the value of variable i.e
- The Program looks at the value of decision variable.
- The program jumps directly to

matching case label.

→ The statements following the case label are executed.

→ Special features of switch statements:

- The "Break" command at the end of a block of statements will make the program jump to end of the switch.
- The program executes the statements after the "default" label if no other cases match the decision variable.

→ The main advantages of this is that the it over a sequence of if-else statement is that it is much faster.

→ Jumping of blocks of code is based on a lookup table instead of a sequence of variable comparisons.

→ While in Nested - if else statements we show how if statement is if-else can be nested inside each other to create more complex paths through a program.

Q5 (a) Differentiate b/w 'Rational operator' and Rational expressions.

(b) Draw flow chart for while loop and Nested if statement. Difference b/w

Rational operator

Rational operators is used to check the relationship between two operands for example:

// checks if a is greater than b
a > b;

Here, [>] is a relational operator. it checks if [a] is greater than [b] or not

→ If the relation is true, it returns 1 where as if relation is false. it returns 0.

Rational expressions:

Rational expressions are defined for use as boolean arguments to procedures and for use with conditional expressions

The rational operators are:

"= ="

< true > if expression are equal.

"<"

">"

<true> if the LHE is greater than RHE. ch-

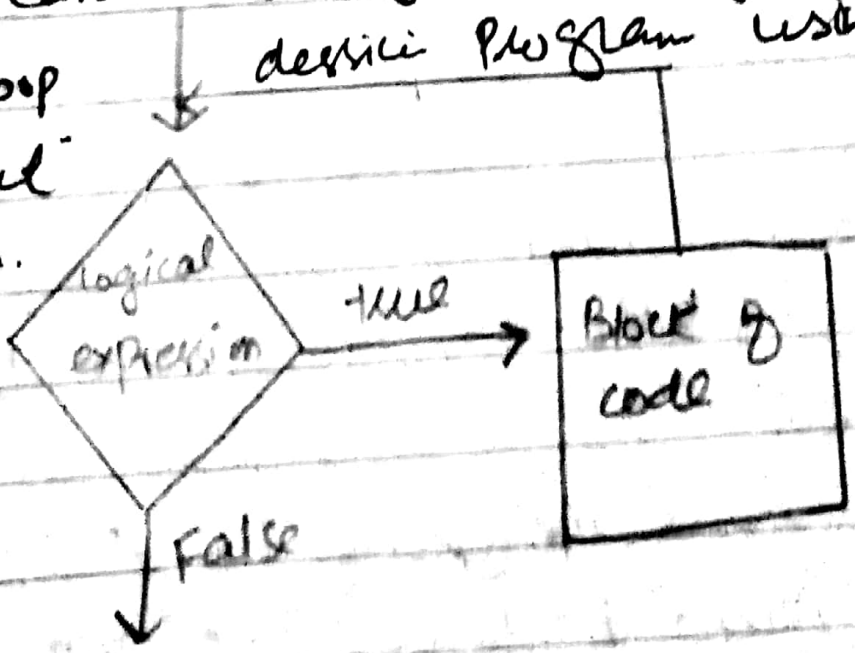
Rational operator.

operator	meaning	example.
==	is equal TO	3 == 5 gives us false.
!=	Not equal TO	3 != 5 given us true.
>	greater than	3 > 5 false

etc.

(b) flow chart for while loop

we can visualize the Programs decisi Program using while loop flow chart diagram.



Statements

(b) Program which Perform the arithmetic operation using arithmetic operators:-

The c program to Perform basic arithmetic operations which are addition, Subtraction, multiplication and division of two numbers. Numbers are assumed to be integers as will be entered by a user. In c language when we divide the two integers we get an integer as a result.

C programming code

```
#include < stdio.h >
int main ()
{
int first, second, add, subtract,
multiply;
float divide;
printf (" enter two integers \n");
scanf ("%d %d", &first, &second);
add = First + second;
Subtract = first - send;
```

multiply = first * second ;

divide = first / (float) second ;

// typecasting

printf ("sum = %d\n", addj);

printf ("Difference = %d\n", subtract);

etc.