Home Work 11

The problems in this problem set cover lectures C13 and C14

1. a. Define a robust algorithm to carry out integer division using repeated subtraction. Your algorithm accepts two integers and returns the quotient and the remainder. Hint: What are the preconditions and postconditions of your algorithm?

Preconditions: Two integers $x,y$

$y$ is non-zero

Algorithm:

Set $R$ to $\text{abs}(x)$
Set $Q$ to zero
While $R \geq \text{abs}(y)$
    Increment $Q$
    $R := R - \text{abs}(y)$
If either $x$ or $y$ are negative
    If both $x$ and $y$ are negative
        Set $R$ to $-R$
    else
        if $x$ is negative
            Set $R$ to $-R$
        Set $Q$ to $-Q$
Display $Q$ and $R$

Postconditions:

$Q$ contains the quotient
$R$ contains the remainder
$x = Q \cdot y + R$, $\text{abs}(R) < \text{abs}(Q)$
b. Implement your algorithm as an Ada95 program, using exception handling to provide robustness.

```ada
1. ----------------------------------------------------------
2. -- Procedure to carry out robust division
3. -- Programmer: Jayakanth Srinivasan
4. -- Date Last Modified : April 17,2004
5. ----------------------------------------------------------
6. with Ada.Text_Io;
7. with Ada.Integer_Text_Io;
8. use Ada.Text_Io;
9. use Ada.Integer_Text_Io;
10. procedure Robust_Division is
11.   X, Y, Q : Integer;
12.   R : Integer;
13.   Divide_By_Zero : exception;
14. begin
15.   loop
17.     begin
18.       -- get the dividend (X)
19.       Ada.Text_Io.Put("Please Enter the X :");
20.       Ada.Integer_Text_Io.Get(X);
22.       -- get the divisor (Y)
23.       Ada.Text_Io.Put("Please Enter the Y :");
24.       Ada.Integer_Text_Io.Get(Y);
25.       Ada.Text_IO.Skip_Line;
26.       if Y = 0 then
27.         raise Divide_By_Zero;
28.       end if;
29.       --set the remainder to absolute value of X
30.       R := abs(X);
31.       -- set quotient to zero
32.       Q := 0;
33.       -- while remainder is greater than absolute value of y
34.       while R >= abs(Y) loop
35.         -- deduct absolute value of y from the remainder
36.         R := R - abs(Y);
37.         -- increment the quotient
38.         Q := Q + 1;
39.       end loop;
40.       --ensure that the sign on the quotient is quotient
41.       if (X<0) or (Y<0) then
42.         if (X<0) and (Y<0) then
43.           -- if both x,y are negative then remainder is negative
44.           R := -1*R;
```
else
if (X<0) then
-- if X is negative then remainder is negative
R := -1*R;
end if;
-- if either x or y not both, then quotient is negative
Q := -1*Q;
end if;
end if;
-- Display the quotient
Ada.Text_Io.Put_Line(Integer’Image(Q));
-- display the remainder
Ada.Text_Io.Put_Line(Integer’Image(R));
-- if the program has reached this part, there were no exceptions
exit;

exception
when Data_Error =>
Ada.Text_Io.Put_Line("Trying to enter a non-integer");
when Divide_By_Zero =>
Ada.Text_Io.Put_Line("Trying to divide by zero");
when others =>
Ada.Text_Io.Put_Line("Dont know what this exception is");
end;
-- this is the end of the block created by the begin statement
end;
-- this is the end of the loop
end loop;
end Robust_Division;

88 lines: No errors
2. a. What is the cyclomatic complexity of the code fragment shown below?

```plaintext
loop
  exit when Flag := True;
  if A < 100 and B > 200 then
    if A > 50 then
      Sum := Sum +2;
    else
      Sum := Sum +1;
    end if;
  else
    if B < 300 then
      Sum:= Sum -1;
    else
      Sum := Sum -2;
    end if;
  end if;
end loop;
```

Hint: Draw the control flow graph

11 Nodes, 14 edges => Cyclomatic complexity = 5.
b. What is the minimum number of test cases needed to test the fragment of code shown below? Justify your answer.

1. if $A < 100$ and $B > 200$ then
2.  
3.  
4.  
5.  
6.  
7.  
8.  
9.  
10.  
11.  
12.  
13. end if;

<table>
<thead>
<tr>
<th>Test Case</th>
<th>A</th>
<th>B</th>
<th>Line Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$50 &lt; A &lt; 100$</td>
<td>$B &gt; 200$</td>
<td>Sum:=Sum+2</td>
</tr>
<tr>
<td>2</td>
<td>$A \leq 50$</td>
<td>$B &gt; 200$</td>
<td>Sum:=Sum+1</td>
</tr>
<tr>
<td>3</td>
<td>$A \geq 100$</td>
<td>$B &lt; 300$</td>
<td>Sum:=Sum-1</td>
</tr>
<tr>
<td>4</td>
<td>Any Other combination of $A$ and $B$</td>
<td></td>
<td>Sum:=Sum-2</td>
</tr>
</tbody>
</table>