PROBLEM SOLVING TECHNIQUES:
THE FIVE STEPS IN CREATING PROGRAMS

STEP 1

Define The Problem: Your definition may range from a few sentences to hundreds of words. A detailed description of what you are trying to accomplish will help you understand the problem better. Try to indicate what information will need to be entered and what information will be output.

STEP 2

Design A Solution:

a) User Interface Design
- how many forms
- what will be in the forms (rough sketch)
    - what data will you need as input
    - what controls (command buttons, etc...) will be required

b) Form Settings

Control Object Type	**Property**	**New Setting**	**Purpose**
Form	Caption	Temp Conversion	Main form
	Name	FrmTemp	
	Borderstyle	3-fixed double	
Command Button	Caption	Convert	Invokes temp conv
	Name	CmdTemp	

c) Flowcharts of event and general procedures using Chartist

STEP 3

Code The Solution Using VB:

STEP 4

Test/Debug The Program:

a) Error handling
    - types of errors (syntax/run-time/logic)

b) Debugging tools
    - setting breakpoints
    - stepping
    - debug window (debug.print)
    - using watch

STEP 5

Document The Program:

a) General specifications for project
- General Info
    - Application Name
    - Date Created
    - Programmer(s)
- List of Filenames
    - Storage Location (directory)
    - Project File (MAK)
    - Global File (GLOBAL.BAS)
    - Form 1 (FRM)
    - Form 2 ...

a) Variable dictionary
- local/module/global variables

b) Program comments

Flowcharts

Flowcharts show the steps and structure of an algorithm. They are made of symbols connected by lines. The symbols contain instructions. The lines are called **flowlines.** Flowlines indicate the order in which the instructions are to be followed. Flowcharts are not only used in computer programming, but can represent and document many processes. A sample flowchart is shown in Figure 8-7 on page 233.

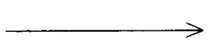
Flowcharts have been used to solve problems for quite a long time. Because they give a clear picture of the program, they are excellent documentation tools. Recently, however, flowcharts have been criticized for a number of reasons. One reason is that they do not lend themselves to top-down development. Flowchart construction begins with the *first* steps in a program, while top-down development focuses on the *major* steps in a program.

Another reason they are criticized is that steps or symbols that are left out are difficult to add later. Making simple changes often means drawing new flowcharts, which takes a lot of time. Large flowcharts can also be confusing and can lose their clarity if they require several pages.

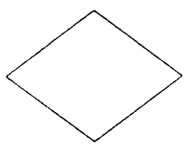
Flowchart symbols

START/STOP **-necessary for every flowchart**

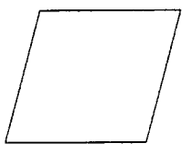
OPERATION **- do something**



ARROW **direction of flow**

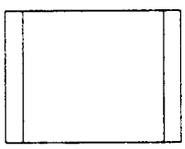


DECISION **answer yes or no**

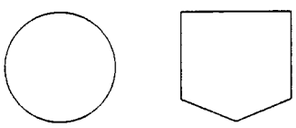


INFORMATION IN **-read box**

INFORMATION OUT **- output box**



PROCESS MODULE **- a module which has been or will be planned elsewhere**



CONNECTOR **- used for longer flowcharts**

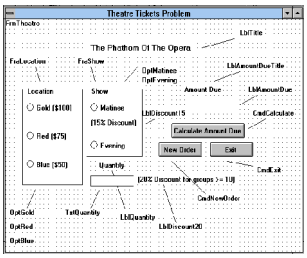
THE PHANTOM OF THE OPERA PROBLEM

STEP 1
DEFINE THE PROBLEM

This application will need to compute the cost of tickets to the Phantom of the Opera show. The user will need to provide the seating location, show time and the number of tickets required. The application will determine the cost of the ticket(s), taking into consideration that a discount will be applied on orders greater or equal to 10, and on matinee shows.

STEP 2
DESIGN A SOLUTION

a) User Interface Design



b) Form Settings

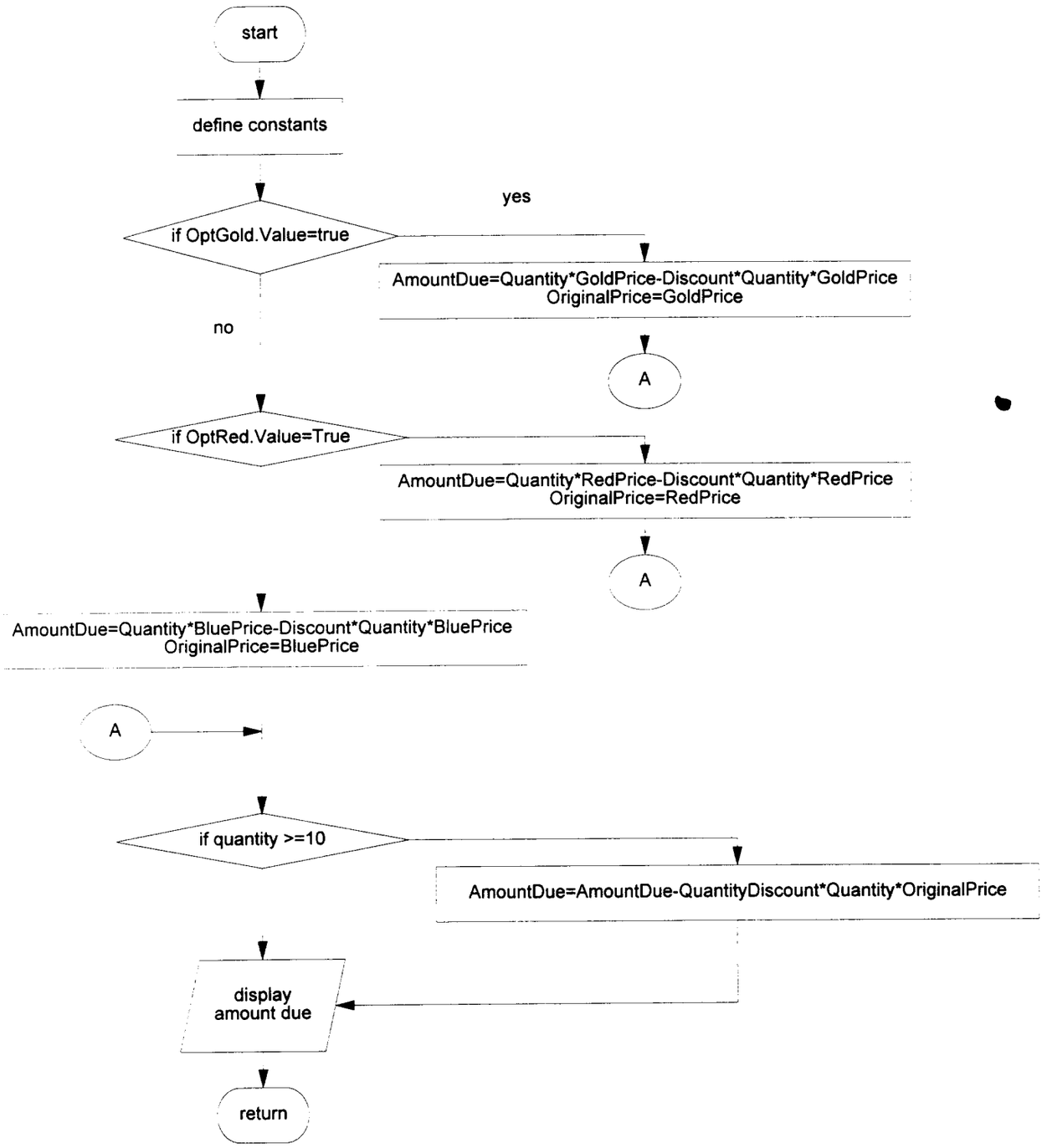
THEATRE.FRM

Control Object type	**Property**	**New Setting**	**Purpose**
Form	Caption	Theatre Ticket Problem	Main Form
	Name	FrmTheatre	
Command Button	Caption	Calculate Amount Due	invokes ticket price calc
	Name	CmdCalculate	
Command Button	Caption	Exit	exits application
	Name	CmdExit	
Command Button	Caption	New Order	clears all user entries and
	Name	CmdNewOrder	answers
Text Box	Text	empty	stores number of tickets
	Name	TxtQuantity	purchased
Frame	Caption	Show	encloses choices for show
	Name	FraShow	time option buttons
Option Button	Caption	Evening	indicates evening show time
	Name	OptEvening	choice
Option Button	Caption	Matinee	indicates matinee show time
	Name	OptMatinee	choice
Label	Caption	15% Discount	label for matinee discount
	Name	LblDiscount15	

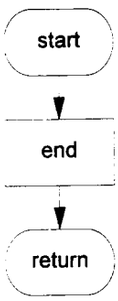
Control Object type	Property	New Setting	Purpose
Frame	Caption	Location	encloses choices for seating
	Name	FraLocation	location
Option Button	Caption	Blue ($50)	indicates blue seat choice
	Name	OptBlue	
Option Button	Caption	Red ($75)	indicates red seat choice
	Name	OptRed	
Option Button	Caption	Gold ($100)	indicates gold seat choice
	Name	OptGold	
Label	Caption	Amount Due	title label for amount due
	Name	LblAmountDueTitle	
	Autosize	True	
Label	Caption	empty	displays amount due answer
	Name	LblAmountDue	
Label	Caption	20% Discount for ...	label for quantity discount
	Name	LblDiscount20	message
	Autosize	True	
Label	Caption	Quantity	label for quantity text box
	Name	LblQuantity	
	Autosize	True	
Label	Caption	The Phantom of the Opera	label for title
	Name	LblQuantity	
	Autosize	True	
	FontBold	True	
	Fontsize	9.75	

STEP 2C FLOWCHARTS

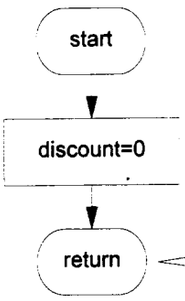
CmdCalculate_Click()



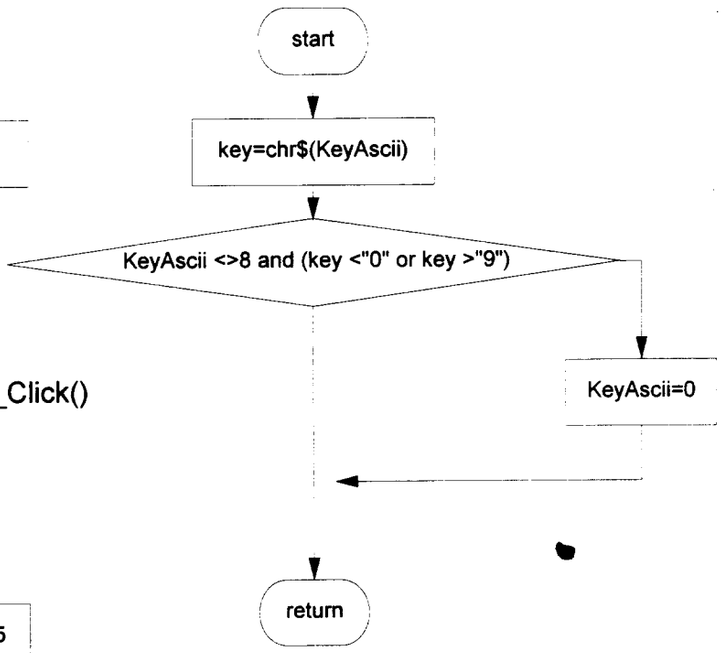
CmdExit_Click()



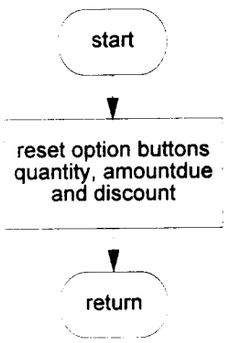
OptEvening_Click()



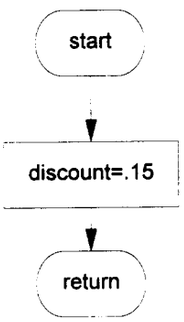
TxtQuantity_KeyPress(KeyAscii as Integer)



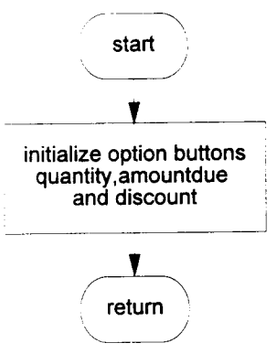
CmdNewOrder_Click()



OptMatinee_Click()



Form_Load()



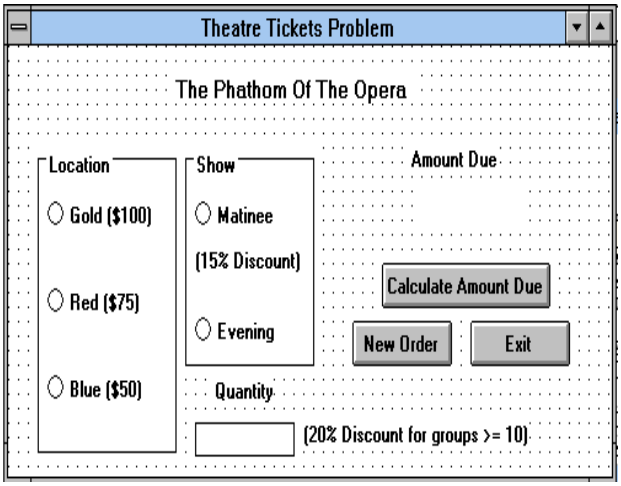
GENERAL SPECIFICATIONS FOR PROJECT

General Info:

Application Name:	The Phantom Of The Opera Problem
Date Created:	Nov /2007
Programmer:	C. Chiarelli

List of Filenames:

Storage Location:	H:\STUDENTS\....
Project File:	TheatreProject
Form 1:	theatrem.cs



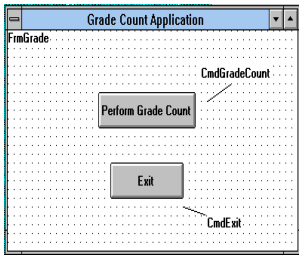
THE GRADE COUNT APPLICATION

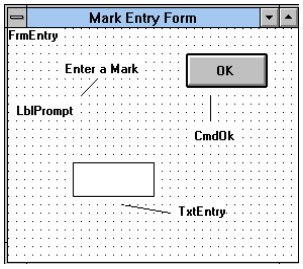
STEP 1
DEFINE THE PROBLEM

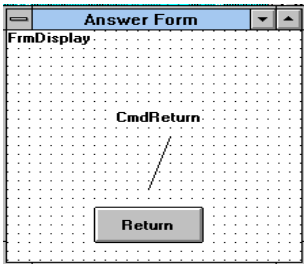
This application will provide individual count totals for five specific grade categories. The user will need to enter a valid test grade between 0-100. The application will determine which grade category it falls into and keep a count of it.

STEP 2
DESIGN A SOLUTION

a) User Interface Design







b) Form Settings

GRADECNT.FRM

Control Object type	**Property**	**New Setting**	**Purpose**
Form	Caption	Grade Count Applic	main form
	Name	FrmGrade	
Command Button	Caption	Perform Grade Count	invokes grade count
	Name	CmdGradeCount	entry and calc
Command Button	Caption	Exit	exits application
	Name	CmdExit	

GRADEANS.FRM

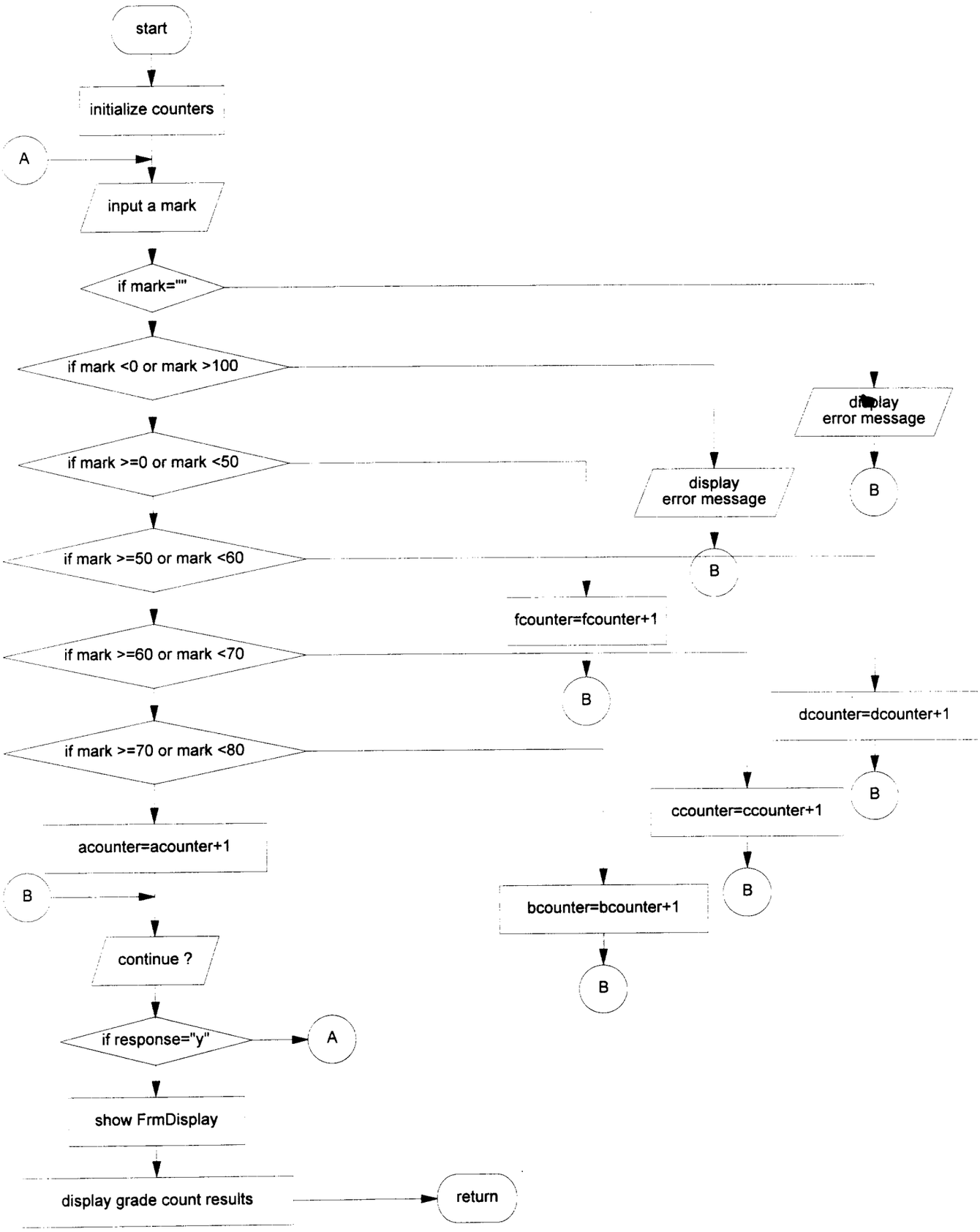
Control Object type	**Property**	**New Setting**	**Purpose**
Form	Caption	Answer Form	displays grade count
	Name	FrmDisplay	results
Command Button	Caption	Return	returns control back
	Name	CmdReturn	to main form

MRKENTRY.FRM

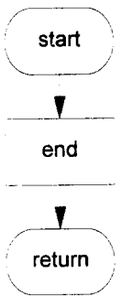
Control Object type	**Property**	**New Setting**	**Purpose**
Form	Caption	Mark Entry Form	marks entered here
	Name	FrmEntry	
	Borderstyle	3-fixed double	
Text Box	Text	empty	stores grade
	Name	TxtEntry	
	Alignment	2-centre	
	Multiline	True	
Command Button	Caption	OK	returns control back to
	Name	CmdOK	CmdGradeCount_Click()
	Default	True	
Label	Caption	Enter A Mark	label for mark entry text box
	Name	LblPrompt	
	Autosize	True	

STEP 2C FLOWCHARTS

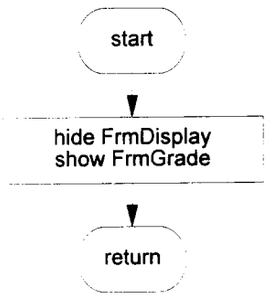
CmdGradeCount_Click()



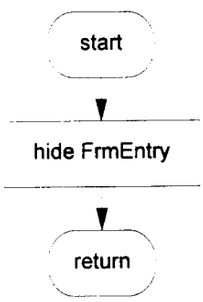
CmdExit_Click() -> FrmGrade



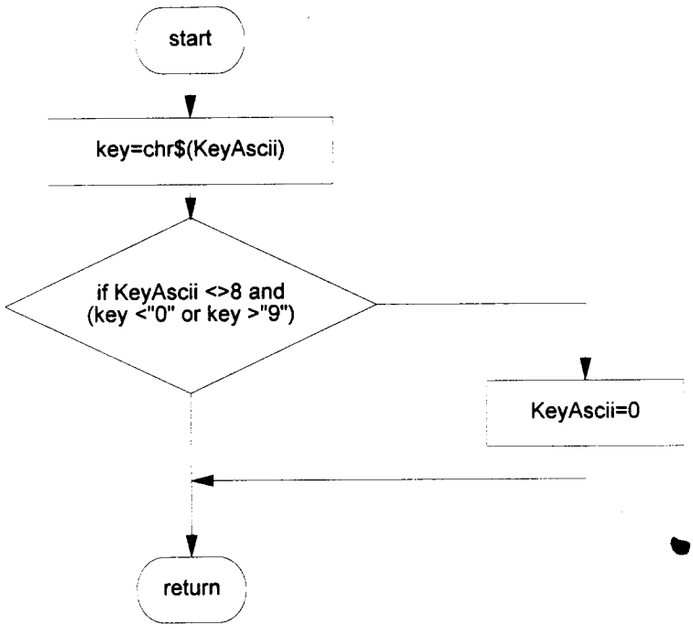
CmdReturn_Click() -> FrmDisplay



CmdOK_Click() -> FrmEntry



TxtEntry_KeyPress(KeyAscii As Integer) -> FrmEntry



GENERAL SPECIFICATIONS FOR PROJECT

General Info:

Application Name:	The Grade Count Application
Date Created:	Nov /2007
Programmer:	C. Chiarelli

List of Filenames:

Storage Location:	H:\STUDENTS\....
Project File:	GradeCountProgram
Form 1:	gradecnt.cs
Form 2:	gradeans.cs
Form 3:	mrkentry.cs