Case 2

Problem (3 points). Apply the Basis path testing technique to obtain the tests cases for the following function. To do so, obtain the associated graph, cyclomatic complexity (using 3 methods) and independent paths (minimize the number of impossible paths but maximize the number of independent paths). Assume class TeamScore exists and it has the necessary properties "Id" and "Score" both of type int. Accessing these properties does not produce any exceptions.

```
int GetWinner(ICollection<TeamScore>? TeamScores, out int MaxScore){
    MaxScore = 0;
    int IdWinner = -1;
    try {
        int l=TeamScores.Count;
        for (int i = 0; i <= l; i++)
        {
            int CurrentScore = TeamScores.ElementAt(i).Score;
            int CurrentId =TeamScores.ElementAt(i).Id;
            if (CurrentScore > MaxScore)
            {
                MaxScore=CurrentScore;
                IdWinner = CurrentId;
            }
        }
    }
    catch {
        Console.WriteLine("An exception occurred!");
    }
    return IdWinner;
}
```

Problem (3 points). Obtain the test cases using the equivalent partitioning technique for a module to calculate the compensation costs caused by flooding. The module receives a text file with records having the following fields:

PersonId: A string of 9 characters where the first 8 characters are digits, and the last character is a capital letter. The letter must be a result of applying a function f to the digits (e.g. f("52644070") is 'Z'

Damages: is a string representing a number of exactly 5 characters between '10000' and '90000' (both included)

Severity: is a string of 3 characters. First character is a letter in the set {'A','M'}. Second and third characters are digits

The module returns:

- -1 if there is any error in the input data
- 10 for Damages between 10000 and 50000 (both included) for any Severity starting with 'A'
- 20 for Damages between 10000 and 50000 (both included) for any Severity starting with 'M'
- 50 for Damages greater than 50000

Problem (4 points). The Ministry of Transportation is hiring ISW to develop software to manage all the activities related to the latest flooding events. The system can be used by any affected person to report damages by providing the location (latitude and longitude), description, keywords and severity level. In addition, if there are any photographs, they may be added and a description for each photograph may be provided when the damage is reported. Pictures may also be uploaded later at any time for a damage report. Based on their expertise, technical assistants may list the existing damage reports (the damage description and severity are listed) and if they select a concrete damage report all the detailed information about the damage is displayed. Based on this detailed information the technical assistant may either close the report (if it is considered that it is not a real damage that needs to be handled) or assign resources to it and a date range for the damage to be solved. Technical assistants must be registered in the system and logged in before any action may be taken. Affected people do not need to register to record their damages. Volunteers may also use the system to help with any activities regarding damages. To do so, volunteers must register and log in. Then, they may obtain the list of existing damages and subscribe as a volunteer to any of them if they consider they have the required skills. Volunteers may also unsubscribe at any other time. Every night the system generates alarms notified to all technical assistants for each damage report that has not been handled for the past 3 days since it was recorded. Finally, when the damage report is not handled in the past 10 days a special type of alarm is generated with an emergency warning signal that is sent to all technical assistants.

Give the previous description, obtain the UML use cases model (Context and structured diagrams) and provide the text template for the Use Case “Report Damage”.

Problem 1

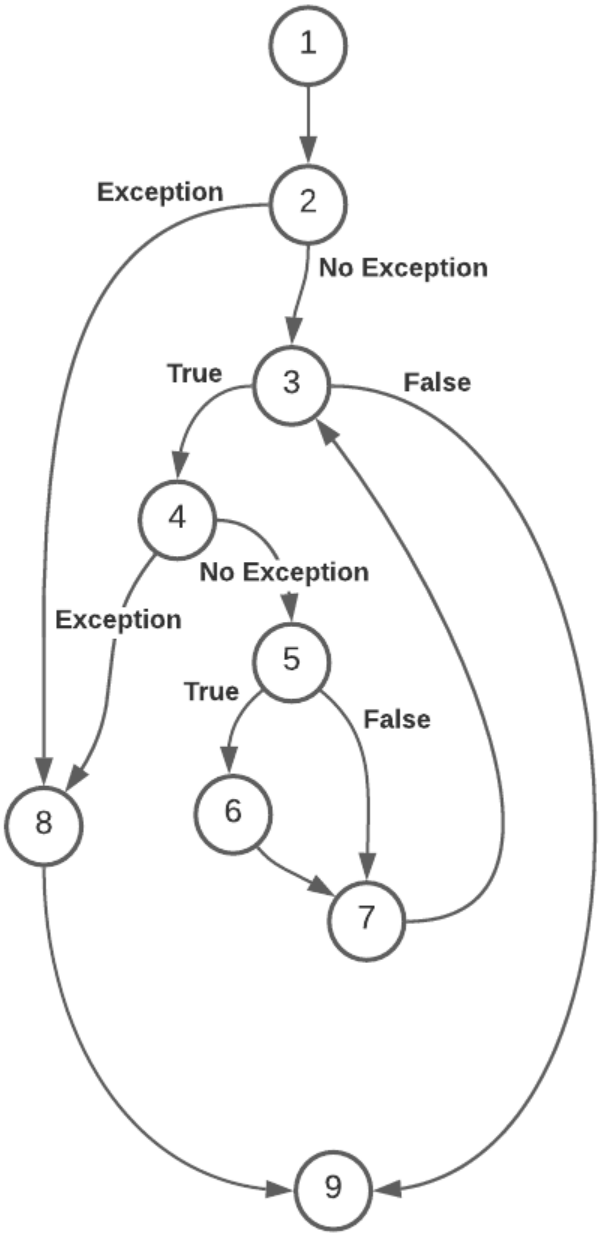
```
int GetWinner(ICollection<TeamScore>? TeamScores, out int MaxScore){

    MaxScore = 0;
    int IdWinner = -1;
    try {
        int l=TeamScores.Count;

        for (int i = 1; i <= l; i++)
        {
            int CurrentScore = TeamScores.ElementAt(i).Score;
            int CurrentId =TeamScores.ElementAt(i).Id;

            if (CurrentScore > MaxScore)
            {
                MaxScore=CurrentScore;
                IdWinner = CurrentId;
            }
        }
    }
    catch {
        Console.WriteLine("An exception occurred!");
    }
    return IdWinner;
}
```

1 2 3 4 5 6 7 8 9



#R= 5
#P+1= 4+1 = 5
#E - #N + 2= 12-9+2=5

Path	TeamScores	MaxScore	Return	Console
1-2-8-9	null	0	-1	An exception occurred!
1-2-3-9	Impossible Path			
1-2-3-4-8-9	[]	0	-1	An exception occurred!
1-2-3-4-5-7-3-9	Impossible Path			
1-2-3-4-5-6-7-3-9	Impossible Path			

But to maximize the number of non impossible paths,

Path	TeamScores	MaxScore	Return	Console
1-2-8-9	Null	0	-1	An exception occurred!
1-2-3-9	Impossible Path			
1-2-3-4-8-9	[]	0	-1	An exception occurred!
1-2-3-4-5-7-3-4-8-9	[{3,0}]	0	-1	An exception occurred!
1-2-3-4-5-6-7-3-4-8-9	[{3,28}]	28	3	An exception occurred!

PersonId: A string of 9 characters where the first 8 characters are digits, and the last character is a capital letter. The letter must be a result of applying a function f to the digits (e.g. f("52644070") is 'Z'

Condition	Valid Classes	Invalid Classes	Heuristic
A string of 9 characters	**(C1)** Strings of 9 characters First 8 characters digits Last character is capital letter Last character function of first 8 digits	**(C2)** Strings less than 9 characters **(C3)** Strings more than 9 characters	Finite Value
First 8 characters are digits		**(C4)** First 8 characters not all digits	Boolean
Last character is a capital letter		**(C5)** Last character not capital letter	Boolean
Character is result of applying f		**(C6)** Last character is not function of first 8 digits	Boolean

Damages: is a string representing a number of exactly 5 characters between '10000' and '90000' (both included)

Condition	Valid Classes	Invalid Classes	Heuristic
String of 5 characters	String of 5 characters String is a number Damages in [10000,90000] [10000,50000] **(C7)**]50000,90000] **(C8)**	**(C9)** Strings less than 5 characters **(C10)** Strings more than 5 characters	Finite value
String is a number		**(C11)** String is not a number	Boolean
Between 10000 and 90000		**(C12)** Damages < 10000 **(C13)** Damages > 90000	Range Reduced classes

Severity: is a string of 3 characters. First character is a letter in the set {'A','M'}. Second and third characters are digits

Condition	Valid Classes	Invalid Classes	Heuristic
String of 3 characters	Strings of 3 characters	**(C16)** Strings less than 3 characters **(C17)** Strings more than 3 characters	Finite value
First character is set {'A','M'}	**(C14)** First character is 'A' **(C15)** First character is 'M'	**(C18)** First character not in set {'A','M'}	Valid accepted values

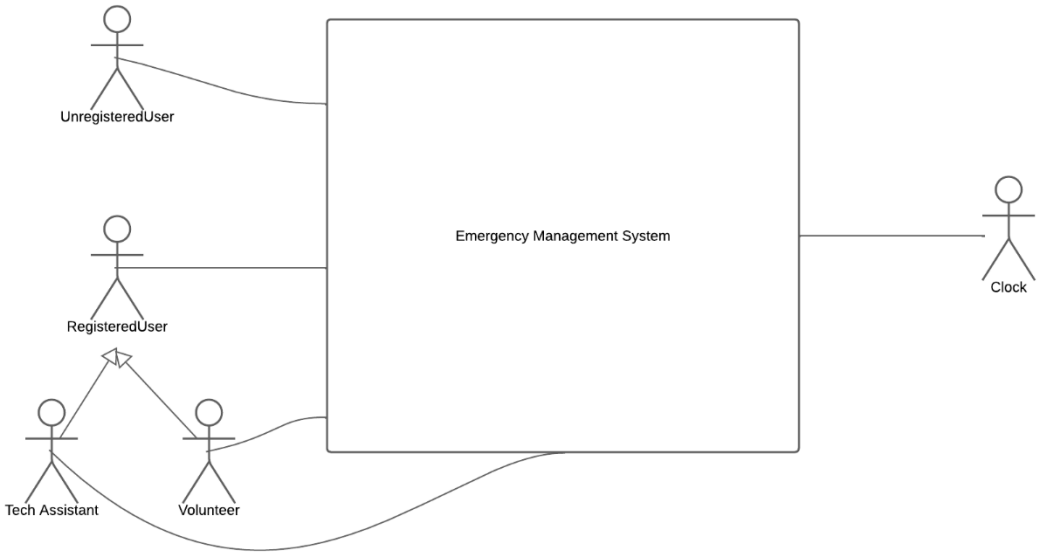
Valid Test Cases

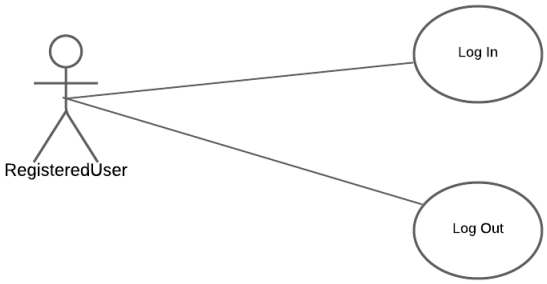
Covered classes	personID	Damages	Severity	Output
C1-C7-C14	52633973Z	20000	A23	10
C1-C8-C15	52633973Z	60000	M43	50

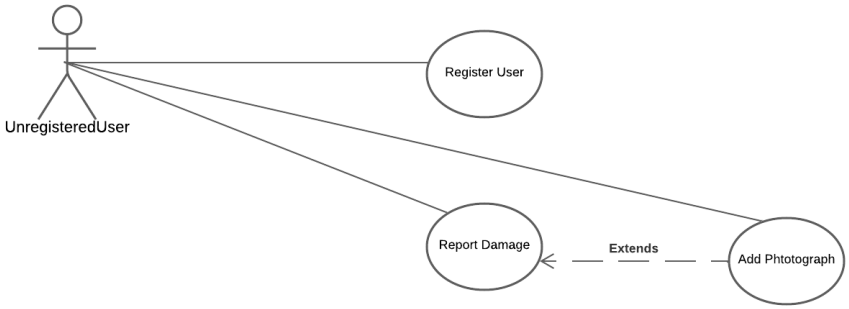
Invalid Test Cases

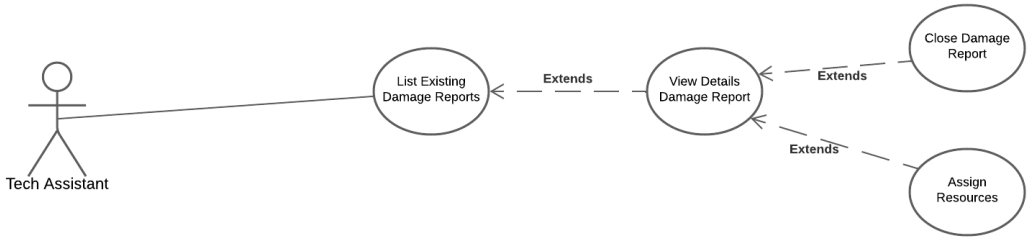
Covered classes	personID	Damages	Severity	Output
C2-C7-C14	Ac	20000	A23	-1
C3-C7-C14	Asdwerqwrte	20000	A23	-1
C4-C7-C14	A1234565Z	20000	A23	-1
C5-C7-C14	12345678!	20000	A23	-1
C6-C7-C14	52633973A	20000	A23	-1
C1-**C9**-C14	52633973Z	2000	A23	-1
C1-**C10**-C14	52633973Z	200000	A23	-1
C1-**C11**-C14	52633973Z	a0000	A23	-1
C1-**C12**-C14	52633973Z	09999	A23	-1
C1-**C13**-C14	52633973Z	91000	A23	-1
C1-C7-**C16**	52633973Z	20000	A2	-1
C1-C7-**C17**	52633973Z	20000	A234	-1
C1-C7-**C18**	52633973Z	20000	S23	-1
C1-C7-**C19**	52633973Z	20000	AAA	-1

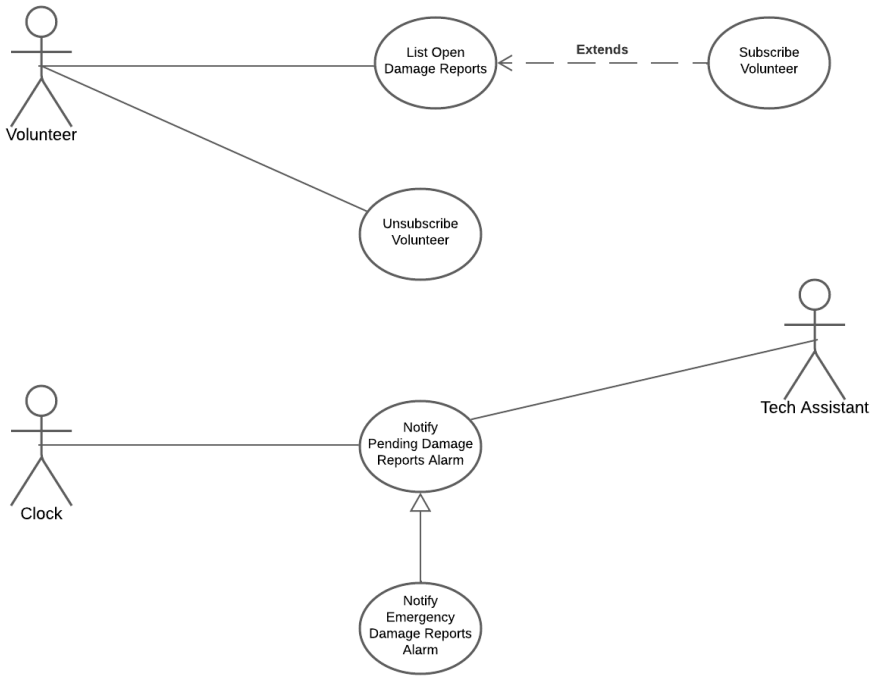
Problem 3


UnregisteredUser
RegisteredUser
Tech Assistant
Volunteer
Emergency Management System
Clock


RegisteredUser
Log In
Log Out


UnregisteredUser
Register User
Report Damage
Extends
Add Phtotograph


Tech Assistant
List Existing Damage Reports
Extends
View Details Damage Report
Extends
Close Damage Report
Extends
Assign Resources



Use Case	Report Damage	
Actor	Unregistered User	
Goal	Report a new Damage	
Summary	Use case to create a new damage report by an unregistered user	
Preconditions		
Postconditions	A new damage report is created	
Includes		
Extends		
Inherits From		
	User Intentions	**System Obligations**
	1. The user states that a damage report has to be created	2. The system requests a location
	3. The user inserts a location	4. The system requests a damage description
	5. The user inserts a damage description	6. The system requests a set of keywords
	7. The user inserts a set of keywords	8. The system requests a severity level
	8. The user inserts a severity level	9. The system validates the data and records a new damage report
Synchronous Extensions		
Asynchronous Extensions	At any moment the user may run use case Add Photograph	