**300583 WSD**

# Web Development Project

**Due: 8pm, Friday 20 October 2017**

This project is to build a footy-tipping website for Australian Football League (AFL) games in 2017. Throughout Australia, footy-tipping is a very popular activity, in which a person tips (i.e., predicts) which teams will win and win by how large margin. A footy-tipping website is the place to hold tipping competitions, in which the person with the closest tips will be selected as the winner.

There are currently 18 teams competing in AFL. Since the actual schedule of AFL games is complicated, this project assumes a simplified schedule. In this schedule, these 18 AFL teams will compete in 20 rounds only, with each round consisting of 9 games. The fixtures of these 20 rounds will be provided to you. We also assume that the results of all games in these 20 rounds are known already, and will be provided to you as well.

The website should be named 'Fantastic AFL Tipping' or other appealing and positive names. In brief, it should provide the following functionality:

- Allowing tipsters (people who tip) to register, login, and make tips for the games in each of the 20 rounds.
- Determining the winner in each round.
- Allowing the administrators of the website to modify tips and look at statistical charts.

The detailed specification for this project is given below. You are suggested to read the entire specification first, and then start with the tasks that are already covered by our online modules.

## 1 Database description

Since the design of the database for the website decides many facets of project implementation, the design of the database is detailed first.

The AFL Tipping website uses an SQL Server database. You should follow our online module 8 to create a local database file for this database, and name it '**AFL_Tipping.mdf**'. This database should consist of the following tables: teams, fixtures, results, tipsters, and tips. The definitions of these tables are given below. For the data types used in the table definition, please refer to the following MSDN article: https://msdn.microsoft.com/en-au/library/ms187752.aspx

### teams

**Table description:** stores all 18 teams in AFL; each team uses one record in this table.

| Field Name | Data Type | Description | Key? |
|---|---|---|---|
| **teamname** | VARCHAR(20) | The name of the team | primary |
| **homestate** | VARCHAR(3) | The abbreviation of the home state for the team, e.g., NSW, VIC, etc. | |

**Notes:** This table should be populated using the 'insert-teams.sql' provided on vUWS. To run an sql query, you should right click 'Tables' in **Server Explorer**, and then click 'New Query'.

### fixtures

**Table description:** stores the fixtures of all 20 rounds; each round uses one record, which consists of the round number, the names of the home and away teams in 9 games respectively.

| Field Name | Data Type | Description | Key? |
|---|---|---|---|
| **roundID** | INT | The round number, ranging from 1 to 20 | primary |
| **home1** | VARCHAR(20) | The name of home team in game 1 | foreign |
| **away1** | VARCHAR(20) | The name of away team in game 1 | foreign |
| **…** | … | … | … |
| **home9** | VARCHAR(20) | The name of home team in game 9 | foreign |
| **away9** | VARCHAR(20) | The name of away team in game 9 | foreign |

**Notes:** This table should be populated using the 'insert-fixtures.sql' provided on vUWS.

## results

**Table description:** stores the results of all 20 rounds; each round uses one record, which consists of the round number, the margins of the 9 games respectively.

| Field Name | Data Type | Description | Key? |
|---|---|---|---|
| **roundID** | INT | The round number, ranging from 1 to 20 | primary |
| **game1** | INT | The margin of game 1 | |
| **…** | … | … | |
| **game9** | INT | The margin of game 9 | |

**Notes:**

- This table should be populated using the 'insert-results.sql' provided on vUWS.
- For simplicity, the results table provided to you only gives the margin of each game. Here, the margin is defined as **`the points of home team – the points of away team`**. For example, if the home team scored 40 points and the away team scored 30 points, then the margin equals 10 points; if the home team scored 20 points and the away team scored 25 points, then the margin equals -5 points. Thus, in addition to the absolute points difference, the margin also indicates which team wins. If the margin is positive, the home team wins; if the margin is 0, the home team draws with the away team; if the margin is negative, the away team wins.

## tipsters

**Table description:** stores all tipsters registered at the website.

| Field Name | Data Type | Description | Key? |
|---|---|---|---|
| **username** | VARCHAR(30) | The username of this tipster. In ASP.NET Identity, this will be the tipster's email address by default. | primary |
| **gname** | VARCHAR(20) | the given name | |
| **sname** | VARCHAR(20) | the surname | |
| **birthdate** | DATETIME | The birth date of this tipster | |
| **address** | VARCHAR(40) | the street address | |
| **state** | VARCHAR(3) | The abbreviation of the state or territory, e.g., NSW, VIC, etc. | |
| **suburb** | VARCHAR(30) | the name of a suburb | |

| | | |
|---|---|---|
| **postcode** | CHAR(4) | 4 digits, we assume tipsters are only from Australia in this assignment |
| **phone** | CHAR(10) | phone number; 10 digits |

**Notes:** This table has no field for password. Tipsters' passwords will be automatically managed by the ASP.NET Identity package, and will not be stored here. After a tipster's registration is successful, this tipster's details other than the password should be inserted into this table.

### tips

**Table description:** stores all tips from all tipsters. Each record stores a set of 9 tips for the 9 games in a certain round from a tipster.

| Field Name | Data Type | Description | Key? |
|---|---|---|---|
| **tipsetID** | INT | The ID of this set of 9 tips; incremented automatically upon record insertion | primary |
| **username** | VARCHAR(30) | The username of the tipster who made this set of tips | foreign |
| **roundID** | INT | the round number | foreign from fixtures |
| **game1** | INT | The margin of game 1 | |
| **…** | … | … | |
| **game9** | INT | The margin of game 9 | |
| **tiptime** | DATETIME | the time when this set of tips is submitted | |

**Notes:**

- For how to make the **tipsetID** field auto incremented in SQL Server, please refer to the second part of: http://www.w3schools.com/sql/sql_autoincrement.asp
- Here, the margin is the predicted margin by a tipster. It is also defined as **`the points of home team – the points of away team`**. Similar to the margin in the results table, the margin here also indicates which team wins. If the margin is positive, the home team wins; if the margin is 0, the home team draws with the away team; if the margin is negative, the away team wins.
- To test the functioning of your webpages, you can manually add some records into this table. However, most records should be inserted into this table by the **MakeTips.aspx** page to be detailed later.

## 2 Two roles of administrators and tipsters

The ASP.NET Identity package can not only maintain users' credentials, but also assign users with different roles. You are required to use this package to divide the users of this website into two roles: **administrators** and **tipsters**. You should follow our online module 6 on how to do this.

Source code should be added to your master page (e.g., Site.Master) such that, after logging in, administrators and tipsters only see links that they have access to (detailed in Section 3 below).

An account with **administrators** role should be created in advance by registering it through the website, and then manually change its role to **administrators**. You should at least create one administrator with username 'admin@AFL-Tipping.com' and password 'Pa$$word1' for your website. After the website starts operating, you should not register new administrators through the registration link of the website. Note that administrators should be maintained automatically by the ASP.NET Identity package, so there is no table for administrators in the **AFL_Tipping** database.

When a new user registers through the registration link of the website, he/she should be assigned the role of **tipsters** automatically. You should add source code to the code-behind of **Register.aspx** to achieve this.

# 3 Navigation and layout

Each page should have a navigation section, with the same 'look and feel' among all pages. This should be achieved by using ASP.NET master pages.

The links contained in the navigation section should be dynamic. The detailed requirements are as follows:

- The links for anonymous users should include the following: Home, Registration, and Login. Note that the Login link here is used by both administrators and tipsters. You should add source code to your project to tell the role of a logged-in user based on this user's username.
- The links for logged-in tipsters should include the following: Home, Fixtures, Make Tips, My Tips, My Ranks, and Logout.
- The links for logged-in administrators should include the following: Home, Modify Tips, Statistics, and Logout.

Except the requirements above, the layout of the website is open to your creativity.

# 4 Homepage — Default.aspx

The Home link mentioned in the above section leads to this page.

- For anonymous users, this page should display a general usage of this website.
- For logged-in tipsters, this page should display an introduction to the links tipsters see.
- For logged-in administrators, this page should display an introduction to the links administrators see.

Hint: Our online module 6 discusses how to display contents based on different roles.

# 5 Tipster registration — Register.aspx

You should work on the existing Register.aspx page, adding appropriate ASP.NET controls to solicit all fields present in the **tipsters** table. Note that the email address of a tipster should be used as his/her username. Validations should be added for all fields as well. The detailed validation requirements are as follows:

- Email – required, in valid email address format (you should add a RegularExpressionValidator for Email into the existing code).
- Password and Retype Password — use existing code in Register.aspx
- Given Name – required, English letters, apostrophe and hyphen only, at most 20 letters long
- Family Name – required, English letters, apostrophe and hyphen only, at most 20 letters long
- Birth Date – required, must be at least 20 years old comparing with the current system date. For example, assume the system date is 09/09/2017; if the tipster was born before or on 09/09/1997, then he passes the validation; otherwise, not. (Hint: You can use a CompareValidator to implement this, setting its ValueToCompare property dynamically by following Module 3. You should find out how to manipulate the DateTime object and the string object in C# by googling it or other means.)
- Address – required

- Suburb – required
- State (must use the DropDownList control) – required
- Postcode – required, 4-digit number only
- Phone – required, in the 0ndddddddd format, where ‘n’ can be ‘2-4’, ‘7-8’, and ‘d’ can be any digit.

After a tipster’s registration is successful,

- The tipster’s details should be inserted into the **tipsters** table; and the tipster should be assigned the role of **tipsters**. (You should add source code to the code-behind file of Register.aspx to achieve these.)
- The tipster should be automatically logged in and redirected to the Default.aspx page. (This is already coded in Register.aspx, so you do not need to do anything about this.)

# 6 Pages for logged-in tipsters

All pages required in this section should be placed in a folder called ‘Tipsters’ in your project. A **Web.config** file should be created under this folder to allow only the logged-in users with the **tipsters** role to access pages in this folder.

## 6.1 Display fixtures — Fixtures.aspx

This page allows a logged-in tipster to look at the fixtures of any round in AFL. This page should first display a dropdown list showing the roundID from 1 to 20. Beneath the dropdown list, there should be a Submit button. After a tipster selects a roundID and clicks the Submit button, the 9 games in that round should be displayed, with each of the 9 games’ information using one row. In each row, the following fields should be presented: the row number (a number between 1 and 9), the name and home state of home team, the name and home state of away team.

You can use either programmatical database access or data-bound controls to implement this page.

**Hints:**

- You need to compose an SQL query that join the **fixtures** table and the **teams** table together for retrieving the information to be displayed.
- To display the fixture of a round below the Submit button, you can use the following approach (other valid approaches are also fine).
  1. Define a  element beneath the Submit button. Inside this , define Label controls corresponding to all pieces of information to be displayed. Set the ‘Visible’ property of  to ‘false’ initially. You can refer to Module 6 for more details about manipulating the ‘Visible’ property.
  2. In the ‘OnClick’ event handler of the Submit button, retrieve the fixture information from the database, assign each piece of information to its corresponding Label controls, and set the ‘Visible’ property of  to ‘true’.

## 6.2 Make Tips — MakeTips.aspx

This page allows a logged-in tipster to make tips for a single round of games in AFL. This page should be implemented with a MultiView consisting of two Views.

### 6.2.1 View 1

The View 1 should display the following:

- A dropdown list showing the rounds for which this tipster has not made tips yet. This list of rounds

should be determined by looking up the **Tips** table. (Hint: The SQL query for this can be composed by using EXCEPT operator or NOT IN operator. Note: By only showing the rounds that have not been tipped, a tipster is only given one chance to tip a round.)

- A ‘Make Tips’ button beneath the dropdown list.

The ‘Activate’ event handler of View 1 should check whether this logged-in tipster has tipped for all rounds. If so, a message should be displayed in View 1 advising the tipster about this, and the dropdown list and the ‘Make Tips’ button should be made invisible.

Upon the click of the ‘Make Tips’ button, View 2 should be activated.

### 6.2.2 View 2

View 2 should display the 9 games in the round selected in View 1, and also present input controls to allow the tipster to make tips for each game. Specifically, View 2 should display each game’s information and the input controls for it using one row. In each row, the following fields should be presented:

1. The row number (between 1 and 9)
2. The name of home team.
3. The name of away team.
4. A DropDownList consisting of 'Win', 'Draw' and 'Lose'.
5. A TextBox for inputting the predicted margin.

Validations on fields 4 and 5 should be also implemented. The validation rules are:

- Field 4 must be selected.
- Field 5 must be an integer between 0 and 200 inclusive. (Notes: Since Field 4 indicates which team will win, a non-negative integer is required for the margin here. Moreover, you are not required to validate that when Field 4 is ‘Draw’, Field 5 should be zero.)

There should be the following two buttons in the bottom of View 2: ‘Submit’ and ‘Select Round’. When the ‘Submit’ button is clicked, all validations in this View should be checked by testing the ‘Page.IsValid’ property. Upon any validation error, the tipster has to correct the error and resubmit his/her tips.

Upon successful validation, the tipster’s username, the roundID, the set of tips for the 9 games and the current system time should be inserted into the **Tips** table. Note that since the tipsetID field has the property of Auto Increment, you do not need to include this field in your insert query. Moreover, because the margins stored in the **Tips** table are different from the margins entered into Field 5, you should do the following conversion on the margins entered into Field 5 before the insertion:

- If Field 4 is ‘Win’, the margin received in Field 5 is untouched.
- If Field 4 is ‘Draw’, the margin received in Field 5 is overwritten by 0.
- If Field 4 is ‘Lose’, the margin received in Field 5 is converted to its negative counterpart.

After successful insertion into the database, an acknowledgement message should be displayed under the ‘Submit’ button.

When the ‘Select Round’ button is clicked, View 1 should be activated, allowing the tipster to select a round ID again.

## 6.3 My Tips — MyTips.aspx

This page allows a logged-in tipster to view the tips he/she has submitted. This page should first check whether the tipster has tipped at least one round. If not, a message should be displayed to advise the tipster about this, and the rest of the page should be kept invisible.

If the tipster has at least tipped one round, this page should display a dropdown list showing the rounds which this tipster has tipped. This list of roundIDs should be determined by looking up the **Tips** table. After a tipster selects a round, the 9 games in this round should be displayed, with each game's information using one row. In each row, the following fields should be presented:

1. The row number.
2. The name of home team.
3. The dynamic text of 'wins', 'draws with' and 'loses to' depending on the result of this game.
4. The name of away team.
5. If the margin is not zero, display " by " + the absolute value of the margin + " points."

Note: This page only displays the tips. No editing is required on this page.

## 6.4 Show Ranks — ShowRanks.aspx

This page allows a logged-in tipster to see the ranks of all tipsters for a given round.

### 6.4.1 The ranking method

Before we detail what to display on this page, we need to specify how the ranks are calculated in this tipping website.

First, we define the **Error Points** of a tip for a game as:

**| the tipped margin for this game – the actual margin of this game |**

In the table below, we give some examples on how to calculate Error Points:

| Tipped margin | Actual margin | Error Points |
|---|---|---|
| -5 | 5 | \| -5 – 5 \| = 10 |
| 5 | -5 | \| 5 – (-5) \| = 10 |
| 0 | 6 | \| 0 – 6 \| = 6 |
| 7 | 3 | \| 7 – 3 \| = 4 |

Then, we define the Error Points for a round by a tipster as the sum of the Error Points for the 9 games in this round by this tipster, and define the Error Points for the entire season by a tipster as the sum of the Error Points for all 20 rounds in this season by this tipster.

With the above said, the ranks of tipsters are obtained by sorting their Error Points in increasing order. The tipster with the least Error Points will be ranked No. 1, and the tipster with the second least Error Points will be ranked No. 2, etc. If there is a tie, the tied teams will receive the same rank. This kind of rank can be obtained by the RANK() method provided in Transact-SQL. For its details, see https://docs.microsoft.com/en-us/sql/t-sql/functions/rank-transact-sql .

### 6.4.2 The specification for this page

This page should first check whether the tipster has tipped at least one round. If not, a message should be displayed to advise the tipster about this, and the rest of the page should be kept invisible.

If the tipster has at least tipped one round, this page should first display a dropdown list showing the rounds for which this tipster has made tips. Similar to **MyTips.aspx**, this list of roundIDs should be determined by looking up the **Tips** table.

After a tipster selects a roundID, the ranks of all tipsters who have tipped that round should be

calculated based on the current data in the database. Then, for each tipster, the following information should be displayed in one row: rank, first name, last name, and Error Points for that round.

**Hint:** The ranking calculation can be done by one SQL query involving the RANK() method, the ABS() method, the '+' and '-' operators, and joining the **Tipsters**, **Tips** and **Results** tables together.

**Note:** Since the calculation of the ranks over the entire season involves very complicated SQL, it is not required in this project.

# 7 Pages for logged-in administrators

All pages required in this section should be placed in a folder called 'Administrators' in your project. A **Web.config** file should be created under this folder to only allow logged-in users with the role of **administrators** to access pages in this folder.

## 7.1 Modify Tips — ModifyTips.aspx

This page allows a logged-in administrator to:

1. List and delete and insert tips.
2. Modify all fields in a record except the primary key '**tipsetID**'.

Specifically, this page should function as follows:

- After clicking the 'Modify Tips' link in the navigation bar, the administrator should see a list of all tips displayed by a ListView control. For each record, the following fields should be displayed: tipsetID, username, roundID, game1, …, game9.
- The editing/deleting/inserting interfaces of the ListView should be enabled.
- Use AJAX controls (e.g., UpdatePanel and ScriptManager) to wrap around the ListView control such that only this ListView is posted back during editing/deleting/inserting, while other parts of the page not.
- For the editing and inserting interfaces, also implement validations for the following fields:
  I. username: required, valid email format
  II. roundID: required, an integer between 1 and 20
  III. game1 - game9: required, an integer between -200 and 200.

In implementing the above validations, please note that:

1. You should use the standard validators such as the RequiredFieldValidator, RegularExpressionValidator, etc. It is not recommended to use the AJAX extenders such as FilteredTextBox here.
2. You should group the validators inside the <InsertItemTemplate> and the <EditItemTemplate> respectively by using the "ValidationGroup" property of the validators. Otherwise, when you click the button, validators in both templates will be invoked. For more details, you can refer to our online module 11.

## 7.2 Plot tipping statistics — Statistics.aspx

This page allows a logged-in administrator to plot two ASP.NET charts on the following statistics.

- The number of tip sets for each round
- The number of tip sets submitted on each week day. (Hint: You can resort to the DATENAME() method in Transact-SQL to achieve this, using the parameter 'weekday'. For more details, you can refer to our online module 12)

This page should first display the following two DropDownLists, which allow the administrator to select the type and dimension for both charts:

- The chart type: Column, Pie, or Line
- The chart dimension: 2D or 3D

When the page is loaded initially, "Column" should be already selected for the chart type, and "2D" for the chart dimension. And both charts with the selected type and dimension should be plotted under the two DropDownLists, with the second chart beneath the first chart. After the administrator changes any selection in the two DropDownLists, both charts should be replotted based on the new selections. (To enable this, the "AutoPostBack" property of the two DropDownLists should be set to "true".)

Moreover, appropriate AJAX controls (e.g., UpdatePanel and ScriptManager) should be used to wrap around the two DropDownLists and the two chart controls such that only these four controls should be posted back while other parts of the page should not.

Finally, for both charts plotted, meaningful chart titles, x-axis and y-axis titles should be included. If a chart type (e.g., a pie chart) does not have x-axis and y-axis titles, then these two titles can be absent.

# 8 Project Administration

## 8.1 Project Groups

This project should be undertaken by a group of 2-3 students (most groups are of 3 students). For how to form groups, you should refer to the UnitOverview.pptx posted on vUWS.

This group work is introduced according to the Unit Outline of 300583/300902. More importantly, when we look at the selection criteria of any advertised jobs, we see almost all of them have one criterion about team work spirit. I hope the group work experience in this unit will give you a solid example in your future job applications, and thus strongly urge you to apply your passion, friendliness and collaboration skills in this group project.

Your group should meet on a regular basis (at least once a week). Each member of the group must provide equal contribution to the project for the group to be successful. It will be the group's responsibility to allocate tasks to each member and to monitor the progress of each member.

## 8.2 Group meetings with tutor

As well as meeting on a regular basis, each project group will also need to meet with your tutor on two occasions during this semester prior to project completion. The purpose of these meetings is to discuss the group and individual progress on the project and to obtain feedback from the tutor. Your tutor will call your group for the meeting in the first 90 minutes of the tutorial class, which is designated for project meetings. All group members should be present for the meeting and be ready by the time the tutor calls your group. If a group member is not present, the tutor will record the absence and continue to hold the meeting with the group members present. The absent group members will receive no marks for the tutor meeting.

### 8.2.1 The first meeting with the tutor

Items to be discussed or marked in this meeting include:

- Task distribution among group members. Each group should write down the task allocation in a 1-page document, and discuss it with your tutor. Tutor should check on the equal allocation.
- All tasks mentioned in Sections 2, 3 and 4 should be done: the creation of two roles, the automatic assignment of a registered user to the role **tipsters**, the dynamic links in Site.Master (the linked pages can be empty), and the dynamic contents in Default.aspx. Since all these tasks are basic ones that server as the starting points of this project, they together account for the 1 mark of this checkpoint. Any incompletion will result in a zero mark.

### 8.2.2 The second meeting with the tutor

Items to be discussed or marked in this meeting include:

- Progress on the project. Each group should write down the progress of each member in a 1-page document, and discuss it with your tutor. Tutor should check whether the progress can ensure the timely completion of the project.
- The populated AFL_Tipping database. All database definitions in Section 1 should be implemented. All '.sql' files provided should have been executed successfully. Moreover, there should be at least 5 tipsters and 12 tip sets added in valid format into the database. This is to ensure that there are sufficient numbers of records for testing and marking purposes. This accounts for the 1 mark of this checkpoint. Any incompletion will result in a zero mark.

## 8.3 Confidential Self and Peer Assessment

To push each group member to contribute equally toward the project, each student will be asked to submit a confidential peer assessment form, in which you can rate each group member's efforts by a percentage. When we give project mark to a group member, we multiply the group project mark with the average of the percentages he/she received (see the Subsection 8.5.2 Allocation of project mark to each group member below for details).

## 8.4 Items to submit for the Project

By the project due date, each student must submit the **confidential self and peer assessment form** through the submission link in vUWS.

Each group must submit:

- ASP.NET Web Forms Project files.
- Readme.txt. This plain text file should provide the following info to markers:
    - What is the starting page of your website?
    - What are the usernames and passwords required to access the web website for the users and administrators respectively?
    - Who are the group members?
    - Special instructions that need to be taken into consideration when marking
- Completed declaration document (download at vuws).

Only one member of the group needs to do the submission for the above items. The submission steps are as follows:

1. Place the Readme.txt and the completed Declaration Document in the root folder of your project.
2. Zip the entire project into a zip file, and name it 'sid1_sid2_sid3.zip', where 'sid' is the student ID of a group member.
3. Submit the zip file:
    - In vUWS, go to Web Development Project.
    - Click on Project Submission
    - Attach your zip file and submit

Note: You can submit the project multiple times until the due date and time (work submitted after the due date will incur late penalties at the rate of 10% per day. Any such late penalty will be applied to all members of the group).

## 8.5 Project Marking

The project will be marked using Visual Studio 2017. Ensure that you have copied your project code to another location on your local system and tested its portability before submitting it.

### 8.5.1 Marking Rubric for Project

The marking rubric is available on vuws via My Grades.

### 8.5.2 Allocation of marks to each group member

As indicated on the WSD Learning Guide, 2 marks are for the two tutor meetings, and 23 marks are for the project. The mark obtained for the project will then be adjusted for each group member based upon the average rating this member receives from all members. The application of this average rating is demonstrated in the following example:

Suppose Group S has 3 members (Student 1, Student 2, Student 3), and the following marks are achieved:

Group Project mark = 20 out of 23,

Student 1 tutor meeting mark = 1 out of 2

Student 2 tutor meeting mark = 2 out of 2

Student 3 tutor meeting mark = 1 out of 2

Student 1 provided the following confidential assessments on members' individual contributions:

Student 1 (self) 90/100,

Student 2 100/100,

Student 3 60/100.

Student 2 provided the following confidential assessments on members' individual contributions:

Student 1 80/100,

Student 2 (self) 100/100,

Student 3 50/100.

Student 3 provided the following confidential assessments on members' individual contributions:

Student 1 80/100,

Student 2 100/100,

Student 3 (self) 60/100.

The average ratings derived from the above confidential peer assessments are therefore:

Student 1 = (90+80+80)/3=83.3

Student 2 = 100

Student 3 = 56.6

Allocated individual marks for the project are therefore derived as:

Final Project Mark = Average Rating × Project Mark + Individual tutor meeting mark

For each of the students in Group S, the Final Project Mark is calculated as below:

Student 1, 83.3/100 x 20 + 1

Student 2, 100/100 x 20 + 2

Student 3, 56.6/100 x 20 + 1