CGI and Basic Web Interfacing

Due: December 3, 2019 on myCourses at 23:55

Lab J will provide some background help for this mini assignment.
Week 10, slides 28 to 44 will help you with CGI development.

YOU CAN WORK IN A TEAM OF TWO, if you want, for this assignment. Both team mates will hand in the same files.

QUESTION: Web Game

We will build a simple multi user web dungeon crawler-like game.

You or your team must build it cooperatively with your classmates.

You can only use HTML, CGI and C. No CSS. No JS, etc.

The goal of the game will be to get 100 gold pieces.

You start the game with 10 gold pieces.

Each dungeon room you enter results in a win of 10 gold pieces or a loss of 5 gold pieces. You will gain or lose gold by engaging in a Q/A activity. Each room will have either a single COMP206 question or a single Zelda question. If the player answers correctly they get 10 gold pieces. If they answer incorrectly, they lose 5 gold pieces. When you cannot pay, you die. When you get 100 gold pieces, you win.

This is an equal opportunity dungeon, so you can start from any room. Simply type the URL of anyone's room to start the game. Your room does not need to be pretty. It only needs to be functional. If you want to make it pretty, then please go ahead, but stick to HTML, CGI and C. Have fun with this.

Each room is responsible to (1) show a question, (2) a picture, (3) be able to process the user's response to the question, (4) award 10 gold or (5) remove 5 gold, (6) display how much gold the user has, (7) display a DIE and (8) WIN message.

Each room will have a textbox input field and a submit button. In that textbox the user can either answer the question posed by the room to win 10 gold pieces or lose 5 gold pieces, or they can input a command. Valid commands are: NORTH, SOUTH, EAST, WEST, or GOLD. Each of the directions is connected to another classmate's room. This means that your room must connect to four other rooms. If we do this right, a player will be able to travel to all 250-500 rooms. You only need to make sure that your room connects to four other rooms.

Once the user has 100 gold they win. When a user presses submit with 0 gold a lose message is automatically displayed. If the user has some gold, even 1 gold piece, the game validates the attempt. If the person got it correct, then they get 10 more gold pieces, otherwise they lose 5 gold pieces. If they do not have enough gold pieces to pay for the loss, then they automatically die.

This is what you need to do:

STEP 0 – The public_html directory

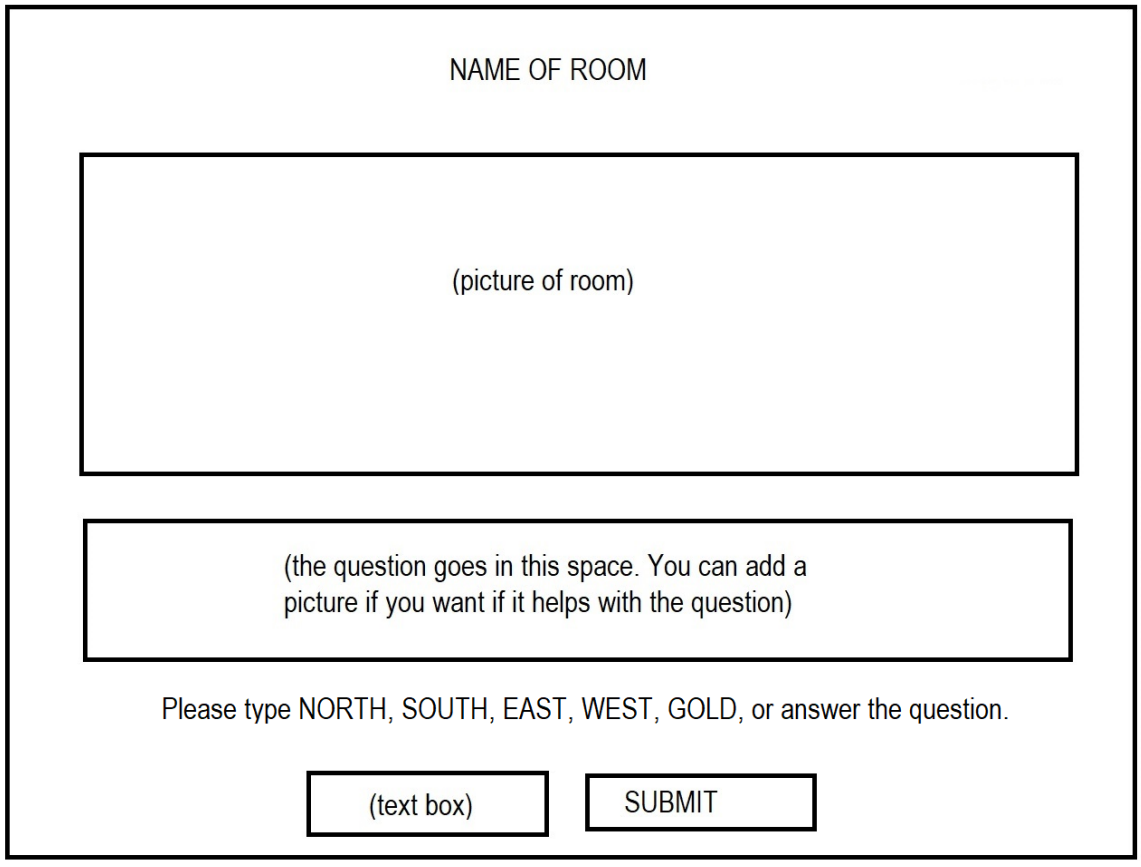
Lab J shows you how to create the public_html directory. Please do this.

You will need to create four files: index.html, answers.c, addgold.c and a picture.

The index.html file will be your dungeon room. The rest of the files are described below.

STEP 1 – Build the room using HTML and CGI

Your dungeon room must look like this:



You can center your NAME OF ROOM with the tag .

To make the game more fun you must add at least one picture to your page. Use the command:

```
<img src="smiley.gif" alt="Smiley face" height="42" width="42">
```

The <img> tag displays a .gif or .jpg or .png file stored in your public_html directory. Write the filename in the src="filename" attribute and specify how big you want the picture to be using the height and width attributes.

Then write the question. Possible tags you might want to use are: bold, <u>underline</u>.

Then write the instruction: "Please type NORTH, SOUTH, EAST, WEST, GOLD, or answer the question."

All the initial interaction with the room will be through the textbox and submit button. The player will need to enter one of the five commands NORTH, SOUTH, EAST, WEST, or GOLD, or provide the answer to the question in the textbox and then press the submit button. Pressing the submit button will call the program `answer` (from `gcc -o answer answer.c`) to process the player's request.

The commands NORTH, SOUTH, EAST, and WEST are movement commands. These commands cause the player to exit the room and enter another room. You will need to talk to four other teams in our class to get their room URL.

The command GOLD will display the number of gold pieces the player currently possesses.

The player will input their command in the textbox and then press the submit button. The submit button will call the program `answer`. This will be implemented using the <form> tag as covered in class. You may use either the POST or GET method to communicate with the C program.

Your web page can look basic and you will receive full marks.

If you want to take the time to make it fun looking, then please do so.

STEP 2 – The Submit button calls a C program called `./answer`

Create a C program called `answer.c` and compile it to the executable `answer` using `gcc -o answer answer.c`. This program receives two possible strings from your webpage. The first string is one of the five commands "NORTH", "SOUTH", "EAST", "WEST", or "GOLD" in all caps. The second string is the answer to the question. You will extract this string from the shell memory if the form used GET or from STDIN if the form used POST (see lecture notes). It is easiest to check for the command first. If the string is not a command, then it must be an attempted answer to the question.

If the player input the word NORTH, SOUTH, EAST, or WEST, the program will programmatically generate a webpage that hyperlinks to another room. Review from the lecture notes how to printf() to the browser. You will need to do something like this:

```
printf("<a href=\"http://URL\">Press HERE to go North</a>");
```

The above printf() will display on the browser a hyperlink. The player will simply click on the word Press HERE to go North with their mouse to cause the browser

to change to another webpage. The HTML tag to link to another webpage is: `<a href="URL">TEXT TO CLICK</a>`. You will need to do this for each direction: NORTH, SOUTH, EAST and WEST.

If the player input the word GOLD, the program will programmatically generate a webpage that displays the amount of gold pieces the player currently owns and a hyperlink to return the player back to the current room.

If the number of gold pieces is 0 or 100 the program must programmatically generate a webpage that displays "YOU WIN" or "YOU LOSE" without a hyperlink to another webpage. The game is now over. This message can be basic looking, or you can make it fun looking.

STEP 3 – The URL and Gold Pieces

Processing the gold pieces is the hardest part of this assignment. It is probably a good idea to leave this to the end. Get steps 1 and 2 working without gold pieces. Once that works, then add the gold pieces.

We need a special HTML tag to handle the gold pieces. Use:

```
<input type="hidden" name="gold" value="10">
```

The input type "hidden" is an invisible input field. The user cannot see it. The user cannot interact with it. But you can programmatically manipulate it.

Your index.html page should not have the hidden field at the beginning. Do not write it in your index.html file. When a user enters the game for the first time, they will do this by inputting a room URL into their browser to start the game. Your game will assume that a missing hidden field means that the user is starting the game and will assume the player has only 10 gold pieces. So, your index.html file should not contain a hidden field. This will be added later through the answer.c program.

When the user presses the submit button the CGI payload will look something like this:

```
http://URL/answer?command=NORTH
```

or

```
http://URL/answer?command=NORTH&gold=50
```

The payload format will depend on the presence of the hidden field. In the first case the hidden field was not present, so we do not see a gold argument. In the second case the hidden field was present.

When your program attempts to extract the gold argument from the payload and does not find it, the program will assume 10 gold pieces.

When you hyperlink NORTH, SOUTH, EAST, WEST, or back to the current page, you will need to programmatically insert the hidden field with the correct amount of gold into the destination page. Unfortunately, you are not permitted to

edit someone else's webpage because it has been chmod read/execute only. The way to get around this is for the owner of the room to provide a service program. Let us call this service program `gcc -o addgold addgold.c`. Instead of the hyperlinks from STEP 2 directing the player to the index.html page of the next room, the hyperlink will direct the player to the addgold program of the next room. The addgold program will display the webpage and insert the hidden field into the webpage.

The URL to the addgold program works like this:

1. Assume my game webpage is at http://www.cs.mcgill.ca/~username
   The player would type that URL at the browser to start the game.
2. When the player presses the submit button (when they want to input a command or answer the question) the form calls the following URL (let us assume the player wanted to go north):
   http://www.cs.mcgill.ca/~username/cgi-bin/answer?command=NORTH&gold=50
   (Note: all executable programs must be in subdirectory cgi-bin)
3. The answer program will programmatically display a webpage with a hyperlink to the northern webpage using this URL:
   http://www.cs.mcgill.ca/~otherpage/cgi-bin/addgold?gold=50
4. The addgold program will display the room owner's index.html page with the hidden field. We are now at step 2 and we can input a command on their page, which leads us to step 3, etc. until WIN or DIE.

The addgold program does its magic in the following way:

- It will fopen("index.html","rt") and fgets() each line and printf() each line of the file to the player's browser.
- When the printing comes to the <form> tag, it inserts an extra printf() that adds the gold. As in:
  ```
  printf("<input type=\"hidden\" name=\"gold\" value=\"%d\"", goldpieces);
  ```
  Note the following:
  - \" prints out the double quote. This is unfortunate but necessary in C.
  - I am assuming goldpieces is an integer and I am adding it into the output with the %d, but there are other ways to do this (maybe the gold was still a string).
- After the inserted line, it continues to read and write the index.html file until the entire file has been printed to the player's browser. The addgold program terminates.
- The player now sees the new webpage with the hidden gold tag. You have now successfully passed a value from your website to the other website.

That is it.

The rest should be straight forward.

Ask the prof or the TA for help.

IMPORTANT

You must use **mimi.cs.mcgill.ca** to create the solutions to this assignment. You cannot use your Mac command-line, Windows command-line, nor a Linux distro installed locally on your laptop. You can `ssh` or `putty` from your laptop to **mimi**, or you can go to the third floor of Trottier and use any of those labs, to complete this assignment.

WHAT TO HAND IN

Everything must be submitted to My Courses before the due date. Remember that you can hand in your assignment up to two days late but there will be a penalty of 5% each day. After that, your assignment will not be accepted. Please hand in the following:

- A readme.txt file with the names and student ID numbers of your team
- A game.html page with a <a href="to your game"> so the TA can quickly go to your game page.
- The C files: answer.c and addgold.c
- **Do not hand in the executable**
- Zip only the C files.
- Make sure to add comments to your C program.
- **Add your name and student ID number as a comment.**

HOW IT WILL BE GRADED

The TA will click on the hyperlink from game.html to go quickly to your home page.

POINTS AWARDED

The assignment is worth a total of 20 points.

- Question
  - 2 points – follow instructions
  - 5 points – STEP 1
  - 5 points – STEP 2
  - 5 points – STEP 3
  - 3 points – WIN and LOSE condition

GRADING RULES

The following rules are followed by the TA when grading assignments:

- A program must run in order to get a grade (even if it does not run well). If it does not run (does not compile) it will receive a zero. (Make sure to run your programs from mimi.cs.mcgill.ca).
- The TA will grade using the mimi.cs.mcgill.ca server.
- All questions are graded proportionally. This means that if 40% of the question is correct, you will receive 40% of the grade.