COMPUTER SCIENCE 101
LOGIC AND COMPUTING
Spring 2011

Paul Y. Cao **Patterson 205, 289-5960** **pcao@ashland.edu**

Goal:

The goal of this course is to help you develop thinking skills such as abstract reasoning and problem solving. You will also gain broad exposure to the discipline of computer science and basic knowledge of computing. Computing is integral to all disciplines and computers are everywhere. So it is a very important topic. I hope that after taking this course, you will be comfortable with computers and familiar with computer science. Also I want to have fun in this course!

Text and Software:

The main text is *Invitation to Computer Science* 5th *ed*. by *Schneider and Gersting*. The recommended (not required) lab manual for the main text is *Invitation to Computer Science Laboratory Manual: C++ and Java, 4th ed.* by *Lambert and Whaley.* Another required book is *Alice in Action 1st ed.* by *Adams.* We will use freeware provided by our textbook as well as *Alice* in our class. You can obtain a free copy of *Alice* software from www.alice.org (choose *Alice* 2.2 in the download page).

Course Outline:

We will cover most of *Invitation to Computer Science* (skipping chapters 9 and 11-13). I hope to cover chapters 1-4.2 and *Alice* programming for the first exam (see below), chapters 4.3-4.4, 5-8, and 10 for exam 2. The final will be comprehensive. To fully benefit from what we do in class, you need to read the text before and after class. We will also include important computer science topics using supplemental handouts throughout this semester.

Note that this is an introductory computer science course. The focus is on abstract reasoning which is central to computer science. For example, when we are dealing with the software layer in using Microsoft Windows, we don't need to be thinking about the details of how the hardware carries out our instructions. We will also use abstract reasoning to look at the representation of information in a computer and explore the idea of an abstract data type. Further, we will reason abstractly when we develop algorithms to solve problems, and translate those algorithms into pseudo-codes and computer instructions. Additionally, we will employ abstract reasoning when we talk about Boolean logic and circuit design.

The course content will focus on problem solving techniques and binary logic design. Brief introduction of important fields in computer science such as operating system, networks, artificial intelligence, high-level programming, information security, graphics, and databases will also be covered.

Office Hours:

MWF 1:00-1:50, 3:00-4:00, Th: 10:30-11:30

Note: You should feel free to drop by as long as my door is open and I am not in a meeting. I am usually in my office between 9:30 am and 4:00 pm if I don't have a class or a committee meeting. If you want to guarantee that I will be in at a time other than office hours, you only need to make an appointment.

Homework/Labs:

The first two homework assignments are for *Alice* programming. The remaining homework problems are from chapters 1 to 5 because those chapters lay the theoretical foundation of computer science. A list of homework problems is given at the end of this syllabus. They will be collected and graded. I'll make solutions available on ANGEL after they are due. All homework are due before class on the due date and should be submitted using the correct dropbox on ANGEL.

We will also have computing laboratories to help you learn. For each chapter, we will have roughly one computing lab to help dissolve what you learn in lectures. I will use ANGEL as a means of making instructions and data available to you. Lab reports are due before the next class after the lab. The reports should also be submitted using ANGEL dropboxes.

Late work will be assessed with 10% penalty per day, including weekends and holidays. No work will be accepted after being late for more than 2 days.

Web:

Course information is available on ANGEL, the official distant learning website at Ashland University (https://au.angel.ashland.edu/). You should have been enrolled automatically and we will use it quite extensively. You will need to use your account to access ANGEL. I'll post homework and lab solutions, announcements, lecture notes, and other course related information there.

Exams/Grades:

We will have 2 exams and a comprehensive final. The exams are worth 100 points each. You may bring a 3x5 card with notes written on one side to each exam. During the exams you may be required to write and understand simple algorithms, answer short questions, and perform basic computing. Your attendance on exams is required. No make-up exams will be given except in documented cases of emergency. The final is worth 150 points and you can bring a 3x5 card with notes on both sides. Homework is worth 100 points (together) and the labs are worth 100 points (together). We will also have one *Alice* programming project worth of 50 points. I'll drop one lowest lab from your grade. Course grades are based on total points earned. The final letter grade will be given according to the following distribution. Your most up to date grades will be available to be viewed on ANGEL throughout the semester.

Grade	Total Points (% range)	Grade	Total Points (% range)
A	564 - 600 (94% - 100%)	C	438 - 461 (73% - 76%)
A-	540 - 563 (90% - 93%)	C-	420 - 437 (70% - 72%)
B+	522 - 539 (87% - 89%)	D+	402 - 419 (67% - 69%)
B	498 - 521 (83% - 86%)	D	378 - 401 (63% - 66%)
B-	480 - 497 (80% - 82%)	D-	360 - 377 (60% - 62%)
C+	462 - 479 (77% - 79%)	F	0 - 359 (0% - 59%)

Exam Schedule:

Exam 1, Monday, Feb. 28th (tentative)
Exam 2, Wednesday, Apr. 6th (tentative)
Final, Monday, May 2nd, 4:00pm – 6:00pm (definite)

Honor Code:

All work that you hand in must be signed with the Ashland Honor Code according to the student handbook, "I affirm that I have adhered to the Honor Code on this assignment." This pledge should also be typed in any electronically submitted work. Working together on assignments or labs is **not allowed**. Examples of violations of academic integrity include: copy other student's work; collaborate on homework or labs; request assistance online or from other resources other than the instructor, etc. I hope none of this will happen in this course. However, should any academic integrity violation occur, the violator will automatically receive an F for this course and the incident will be reported to the academic integrity council. You should feel free to ask me any question about any assignment or lab.

Schedule:

The weekly schedule is subject to change but I expect to be able to stick to this schedule pretty well. For detailed daily schedule, please refer to the calendar on ANGEL. In the reading list, * means there will be supplemental reading materials. By default, the readings are from the main text. Readings in *Alice in Action* are noted as "in *Alice*" in bold fonts.

Week of	*Topics*	*Reading*
01/10	Introduction / Pseudo-codes	Ch. 1, §2.1-2.2
01/17	Alice introduction / Methods in *Alice* (**MLK holiday**)	Ch. 1-2 **in *Alice***
01/24	Functions, variables, if/else in *Alice*	Ch. 3, §4.1-4.2 **in *Alice***
01/31	Loops and data structure in *Alice*	§4.3-4.4, §5.1-5.2 **in *Alice***
02/07	Algorithm example / Efficiency analysis	§2.3, §3.1-3.4
02/14	Sorting algorithms / Position based numbers	§4.1-4.2
02/21	Boolean logic and gates / Circuit design	§4.3-4.5
02/28	Computer architecture / Introduction to OS (**Exam 1, 2/28**)	Ch. 5, §6.4
03/07	**Spring break**	Enjoy the break!
03/14	Computer networks / Info. Security / HTML intro.	Ch. 7-8, §10.3.2*
03/21	More HTML / JavaScript introduction	§10.3.3*
03/28	More JavaScript (**CAS symposium)**	Extra readings
04/04	E-commerce and database (**Exam 2, 4/6**)	Ch. 14*
04/11	Access 2007 intro. / Artificial intelligence	Ch. 15
04/18	Computer graphics (**Easter holiday)**	Ch. 16
04/25	Computers and society / Final review	Ch. 17

Attendance:

Attendance in this course is required. You are expected to attend each lecture and to be in class on time. In case of an absence, you must provide a documented excuse of absence or get my consent. Excessive (more than 4) unexcused absences will result in a 1% off the final total.

Students with Disabilities:

Students with disabilities who have documentation on file with Classroom Support Services (105 Amstutz, extension 5953) are entitled to reasonable academic accommodations. I strongly encourage you to talk to me as early as possible in the semester so appropriate arrangements can be made.

Assignment 1 (*Alice* basics)

1. Using the **heBuilder** or **sheBuilder** (under **People** in the **Local Gallery)**, build a person. Place the person in a world containing a building. Using the **walk(), move(),** and **turn()** messages, write a program that makes him or her walk around the building. Adjust your camera angel properly such that the people object is always visible.

2. Using the **heBuilder** or **sheBuilder** (or any other persons with enough detail in the Alice Gallery), build a person and add him/her to your world. Using your person, build an aerobic exercise class video in which the person leads the user through an exercise routine. The person must do at least two different kinds of routines. Change the angle of your camera at least once in this video. Manage your world structure properly.

3. Using the **heBuilder** or **sheBuilder (**or any other persons with enough detail in the Alice Gallery**),** build a person and add him/her to your world. For your person, define an object method named **walkInSquare()** that has a parameter named **edgeLength**. When **walkInSquare(dist)** is sent to your person, he or she should walk in a square with edges that are each **dist** meters long. Make certain your person begins and ends at the same spot. When the person is done, have the person say the area and perimeter of the square

4. Build an undersea world containing a **goldfish**. Build a **swim()** method for the **goldfish** that makes it swim forward in a realistic fashion. The swim function should have a parameter that lets the sender of the message specify how much the fish should swim. Add a **shark** to your world, and build a similar **swim()** method for it. Build a program containing scenes in which the **shark** chases the **goldfish**, and the **goldfish** swims to its giant cousin goldfish that chases the **shark** away.

Due Friday 1/28/2011 before 2:00pm

Assignment 2 (Advanced topics in *Alice*)

1. Build a world containing a person who can calculate the average of a sequence of numbers in his or head. Have the person ask the user how many numbers are in the sequence, and then display a **NumberDialog** that many times to get the numbers from the user. When all the numbers have been entered, have your person "say" the average of those numbers.

2. From the **Alice Gallary,** choose a clock that has subparts for the minute and hour hands.
    a. Build a clock method named **run()** that moves the minute and hour hands realistically (that is, each time the minute hand completes a rotation, the hour hand should advance 1 hour). Define a parameter named **speedUp** that controls the **duration** of the hand movements, such that **run(0)** will make the clock run at normal speed, **run(60)** will make the clock run at 60 times its normal speed, **run(3600)** will make the clock run at 3600 times its normal speed, and so on.
    b. Build a clock method **setTime(h, m)** that sets the clock's time to **h:m** (**m** minutes after hour **h**).
    c. Build a clock method **setAlarm(h, m)** that lets you set the clock's alarm to **h:m.** Then modify your **run()** method such that when the clock's current time is equal to **m** minutes after hour **h**, the clock plays a sound (for example, Alice's **gong** sound).

3. Create a "springtime" scene that runs for a minute or so, starting with an empty filed but ending with the field covering with flowers. The flowers should grow out of the ground as your scene plays. Make sure that flowers appear in a different order or pattern every time your program is run. You can use random-number generation to achieve it. Make your program as short as possible by storing the flowers in a list.

Due Monday 2/7/2011 before 2:00pm

Assignment 3 (Algorithms)

1. Trace through the decimal addition algorithm of Figure 1.2 in the textbook using the following input values
   $m = 3$
   $a_2 = 1, a_1 = 4, a_0 = 8$
   $b_2 = 0, b_1 = 7, b_0 = 9$
   At each step, show the values for c_3, c_2, c_1, c_0 and *carry*.

2. Modify the algorithm of Figure 1.2 in the text such that it does not print out a non-significant zero. In its current form, the algorithm will output 0391 if you add 152 with 239. The leading 0 of 0391 is the non-significant zero.

3. One way to do multiplication is by repeated addition. For example, 47×25 can be evaluated as $\underbrace{47 + 47 + \cdots 47}_{25 times}$. Design an algorithm for multiplying two positive integers *a* and *b* using this technique.

4. Develop a formal argument that proves the sequential search algorithm we discussed in our class cannot have an infinite loop. (Hint: you can find out the maximum number of times the algorithm will execute its statements and show that number is finite)

5. For the find the largest number algorithm, answer the following questions
   a. If the number in the list are not unique and therefore the largest number could occur more than once, would the algorithm we discussed in the class find the first occurrence? Last occurrence? Every occurrence? Explain your answer.
   b. One instruction in the algorithm is while ($i \le n$). Explain what would happen if we change that instruction to
      - while ($i < n$)
      - while ($i \ge n$)
      - while ($i = n$)

6. Design an algorithm that is given an input as an integer value $k \ge 0$ and a list of *k* numbers noted as $n_1, n_2, \cdots, n_k$. Your algorithm should reverse the order of the number in the list. That is if the original list has *k* as 5 and contained $n_1 = 5, n_2 = 13, n_3 = 4, n_4 = 20, n_5 = 10$. After your algorithm executes, the values stored in the list will be $n_1 = 10, n_2 = 20, n_3 = 4, n_4 = 13, n_5 = 5$. Please note that your algorithm shouldn't simply display the list backward. The values of the variables in the list have to be changed to reflect the reverse order. (Hint: use a loop to swap values at corresponding positions. Also pay attention to if *k* is even or odd).

7. A tennis tournament has 102 players. A single match involves 2 players. The winner of a match will play the winner of a match in the next round, whereas losers are eliminated from the

tournament. The 2 players who have won all previous rounds play in the final game, and the winner wins the tournament.

a. One question we may ask is what is the total number of matches needed to determine the winner? I have one algorithm that may help determine the answer. Compute 102/2=51 to get the number of pairs (matches) in the first round that results in 51 winners to go to the next round. Compute 51/2=25 with 1 left over, which results in 25 winners, plus the 1 left over, to go to the next round. So the total number of games would be 51+25+… Finish this process to find out what the number of games to be played to determine the final winner.
b. Design an easier algorithm to solve the problem. It involves very minimum calculations. (Hint: each match results in exactly one loser, so there must be the same number of matches as losers in the tournament. And there is only one winner in the whole tournament).

8. Perform selection sort on the following lists. Show the list after each exchange.
   a. 1 2 3 4 5
   b. 5 4 3 2 1
   c. 3 3 3 3 3
   d. 9 0 7 5 2

9. Perform bubble sort on the same lists as in problem 8. Show the list after each exchange.

10. Use binary search to search whether a given key exists in the list. Show all steps including the new search boundaries and the item being compared with the key at each step.
    a. List: Boris, Cathy, Chris, Darren, Iyad, Justin, Gordon, Paul, Tom, Vickie
       - key: Elsa
       - key: Chris
       - key: Vickie
    b. List: 3 6 7 9 12 14 18 21 22 31 43
       - key: 35
       - key: 3

11. An algorithm that is linear takes 10 seconds to execute on a particular computer when given 100 input items. How long would you expect it to take when given 500 input items? Answer the same question if the algorithm is a quadratic algorithm.

Due Friday 2/18/2011 before 2:00pm

Assignment 4 (Binary numbers)

1. Using the idea of positional numbering system, determine the decimal value of the following numbers
    a. 122 (base 4)
    b. 457 (base 8, also called *octal*)
    c. 12AF (base 16, also called *hexadecimal*. A represents 10, B represents 11, C represents 12, D represents 13, E represents 14, and F represents 15)
    d. 4455 (base 6)

2. Determine the decimal value of the following binary numbers
    a. 1100 0
    b. 1001 0011 11
    c. 1111 1
    d. 1001

3. Convert the following decimal numbers into binary
    a. 101
    b. 23
    c. 55
    d. 255

4. Show the step-by-step addition of the following binary numbers, including showing the carry bit to each successive column.
    a. 1000 + 1100
    b. 1111 0001 + 0101 1010
    c. 1110 + 1110
    d. 1110 1 + 1110 1
    e. Can you find any pattern from *c* and *d* where a binary number is basically added to itself (it is equivalent of doubling the number)?

5. Use the ASCII table to show the internal binary representation for the following character strings.
    a. AbC
    b. $25.00
    c. Paul
    d. CS 101
    e. (5+2)=7

6. How many binary digits would it take to represent the following phrase in ASCII code?

    In CS 101, we learned how text is stored inside computers.

7. How many bits does it take to store a 2-minute song using an audio encoding method that samples at 40,000 samples per second and each sample is represented using 16 bits, and does not use any compression method?

8. How many bits does it take to store an uncompressed 1280 x 768 RGB color image? The color depth of RGB is 8 bits for Red, Green, and Blue, respectively.

Due Friday 2/25/2011 before 2:00pm

Assignment 5 (Computer architecture)

1. If *a*=1, *b*=2, and *c*=3. What is the value of each of the following Boolean expressions?
   a. (a>1) OR (b=c)
   b. [(a+b)>c] AND (b≤c)
   c. NOT(a=1)
   d. [NOT(a=b)] OR (b=c) OR (a=c)
   e. (a>b) OR (b>c) OR (c>a)

2. Using AND, OR, and NOT gates to implement the following circuits
   a. NAND logic circuit (stands for *Not AND*)

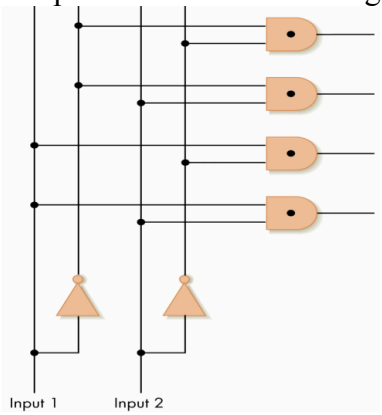
a	b	Output
0	0	1
0	1	1
1	0	1
1	1	0

   b. Logical implication (widely used in logic theory)

a	b	Output
0	0	1
0	1	1
1	0	0
1	1	1

3. Build a **majority-rules circuit**. This is a circuit with three inputs and one output. The value of its input is 1 if and only if two or more of its inputs are 1; otherwise, the output of the circuit is 0. For example, if the inputs are 0, 1,1, your circuit should output a 1. If its inputs are 0, 1, 0, it should output a 0.

4. Given the following circuit which has two inputs and four outputs, give its truth table and try to give an example where this circuit might be used (Hint: treat the two inputs as a binary number).



5. At a minimum, how many bits are needed in the MAR with each of the following memory sizes?
    a. 1M bytes
    b. 10k bytes
    c. 256M bytes
    d. 4G bytes

6. A memory unit that is said to be 640KB would actually contain how many memory cells? What about 512MB? What about 2G bytes?

7. Describe how MAR and MDR are used to find out the value of a specific memory cell? What about storing a value into a memory cell?

Due Monday 3/14/2011 before 2:00pm