CS 4596/CS 6596 Fall 2016
Program #2
Due Wednesday, November 9

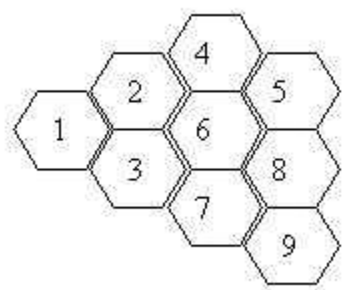
Objectives:

Model the behavior of a Mobile Switching Center (MSC) as it allocates and deallocates channels to connections using a **fixed** allocation scheme.

Details:

This is a simulation. As a result, you need only write a single program which will run on a single machine. You will decide what algorithm the MSC uses to allocate channels.

You should assume 3 cell reuse (cluster size N = 3) and a coverage area of 9 hexagonal cells (3 clusters) which are adjacent. The cells in cluster A are numbered 1,2,3, cluster B are 4,5,6, and cluster C are 7,8,9. See figure below.



The cell radius is 1000 meters. Assume you have a total of 15 voice channels per **cluster**, so channels 1-5 are used in cells 1,6,9, channels 6-10 are used in cells 2,5,7, and channels 11-15 are used in cells 3,4,8. The MSC must use a fixed channel allocation scheme, however, you get to decide how channels within a cell are allocated! The order in which you assign channels in the various cells will have an effect on the signal quality though. The scheme that you use to determine this order is called a **hunting sequence**. For instance, a very simplistic scheme would be to start with the first channel allocated to the cluster (e.g., channel 1 in cluster 1) each time and keep trying channels (2, 3, 4, ...) until you get one that works (is not in use in this cluster and has an acceptable SIR). Assume a blocked calls cleared model.

Assume that the mobiles are not very mobile, rather that they stay in the same cell for the entire length of their call. No handoffs are necessary.

Each time a channel is allocated, use equation 2.8 to calculate the SIR. This formula was given in class and a crude representation is given below. Use a path loss exponent (**n**) of 4.0. We require an estimated

SIR of 22 dB to complete a call. If the SIR will be less, the call must be refused. We will only consider **co-channel** interference; you may ignore adjacent channel interference. Note that you must convert SIR to dB before comparing to the required value. Remember that SIR (in dB) = 10log(S/I).

Eqn 2.8: S/I = (R^-n) / (sum over all interferers i ((D sub i) ^-n))

Note also that equation 2.8 takes into account the relative distance of the co-channel cells. To make this easy, assume the distance is in whole cell diameters. So an interfering cell is either 1, 2, or 3 diameters (2000, 4000, or 6000 meters) away. You should manually construct a table in advance to provide this distance information. Here is the top half of the table:

To/From	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5	Cell 6	Cell 7	Cell 8	Cell 9
Cell 1	0	2000	2000	4000	6000	4000	4000	6000	6000
Cell 2		0	2000	2000	4000	2000	4000	4000	6000
Cell 3			0	4000	4000	2000	2000	4000	4000
Cell 4				0	2000	2000	4000	4000	6000
Cell 5					0	2000	4000	2000	4000
Cell 6						0	2000	2000	4000
Cell 7							0	2000	2000
Cell 8								0	2000
Cell 9									0

For example, if you were trying to assign channel 12 to a new call in cell 3, and that channel was already used in cell 4 (but not in cell 8), then there would be 1 co-channel interferer which was 2 cell diameters (4000 meters) away. The formula would work out to:
S/I = (1000^-4)/(4000^-4) = 256 = 24 dB. So channel 12 could be successfully allocated to the call in cell 3.

If you tried to assign channel 7 in cell 2 (cluster A) and that channel was already used in cell 4 (cluster B) and 7 (cluster C), then there would be 2 co-channel interferers which are 1 cell diameter (2000 meters) and 2 cell diameters (4000 meters) away respectively. The formula would give: S/I = (1000^-4)/[(2000^-4) + (4000^-4)] = 15 = 11.7 dB. So channel 7 could not be allocated - another channel would have to be tried.

Note, if there are 2 co-channel interferers, one 4000 meters away and the other 6000, the denominator is (4000^-4 + 6000^-4), NOT (4000+6000)^-4

You must track 2 statistics as your program runs: the **Grade of Service** and the **average SIR** for successful calls. The **Grade of Service** (GOS) being provided by this system is simply the %blocked calls and can be calculated at completion of the simulation by dividing the number of blocked calls by the number of call attempts. The average SIR for successful calls can be calculated by adding all the SIR values for calls which succeed and dividing by the number of successful calls at the completion of the simulation.

You may write your program in any language (C, C++, Java, Python, SimScript II, etc.) and run it on any machine. You don't need to demo your program (although clearly it must run correctly.)

I suggest using an **event-driven simulation**. This means that you build an event queue (a simple linked list ordered by time.) Read all the call attempts in and place them in the queue. Then take an event off the head of the queue and process it. This means, decide if a channel can be allocated, using your channel allocation scheme. If not, discard this event. If so, place a corresponding disconnect event in the queue at the appropriate time = current time + call duration (note that it probably will not go at the end of the queue). Now repeat until the queue is empty.

Requirements:

Your program must read in an ASCII file describing the time of origination, the cell, and the length of connection for the mobiles in your MSC's coverage area. You will be supplied with this file. The format is described below.

Format of input file:

Number Time Cell Duration

Where Time is in seconds from the beginning of time (0 seconds), Cell is a number from 1-9, and Duration is in seconds. Tabs are used to separate entries. Times will be given in increasing order. There will be duplicate times though (multiple events which occur during the same second.) Handle these in a first come-first serve fashion.

Example input file:

1 2 3 60
2 3 4 120
3 4 4 40
4 4 6 80
5 10 2 100

Your program must generate some output for each call attempt, logging the call number, time, cell, duration, calculated SIR, whether the call was accepted or not, and if so, which channel was allocated to it. For each accepted call, also print a list of the cell/distance pairs for all co-channel interferers (on the selected channel). For rejected calls, print a list of channel/reason pairs which shows why no channel was available. Acceptable reasons are: In Use, Low SIR, and possibly others, depending on your allocation strategy. For the Low SIR reason, be sure to print out the calculated SIR. When a call has disconnected, print out the call number, call start-time and end-time, cell, and channel it was using. When all calls are disconnected, your program must write out the total number of calls accepted, the number blocked, the GOS for the system, and the average SIR for accepted calls. **Note:** If there are no interfering co-channel cells when you calculate the SIR, use 35 dB for the SIR calculations. Make sure that your log entries are in columns so that they can be read easily (unlike this web page.)

Format of output file:
For new calls:

New Call: Number = a Time = b Cell = c Duration = d Accepted/Rejected, Channel=x, SIR=y (no channel or SIR for rejected calls)
Interferers: Cell /Distance Cell/Distance ... (for accepted calls only)
Reasons: Channel/Reason Channel/Reason ... (for rejected calls only)

For calls disconnecting:
Disconnect: Number = a StartTime = b EndTime = c Cell = d Duration = e Channel = f

Example output file:

```
New Call: Number = 1 Time = 2 Cell = 3 Duration = 60 Accepted, Channel = 5, SIR = 35 dB
Interferers: None
New Call: Number = 2 Time = 3 Cell = 4 Duration = 120 Accepted Channel = 5, SIR = 24 dB
Interferers: 3/4000
 ...
New Call: Number = 3 Time = 4 Cell = 4 Duration = 40 Rejected
Reasons: 1/In Use 2/In Use 3/Low SIR=19 dB 4/In Use 5/Low SIR=12 dB
 ...
Disconnect: Number = 1 StartTime = 2 EndTime = 62 Cell=3 Duration = 60 Channel=5 …

Totals: 150 calls accepted, 5 calls rejected, 3.33% GOS, Average SIR = 23.2 dB
```

Your program must be well-commented including function headers and explanations of complex code. Proper indentation is expected.

Tips:

Consider your **data structures** carefully. They will make all the difference. You might define the following structures:
The event queue
The distance matrix (filled in by hand)
A channel usage matrix - this is the important one as you need to be able to determine the status of, for example, channel 2 in all three clusters easily when attempting to serve a new call.

To hand in:

- A printout of your commented code
- A printout of your program's output using the high-load input file (posted separately).
- A printout of your program's output using the low-load input file (posted separately).
- An explanation of your results. Describe:
    a) Your channel allocation strategy (hunting strategy)
    b) For each run, whether you thought the GOS and average SIR were reasonable (and why).
    c) Did the system behave the way you thought it would? Explain why.
    d) How could this system be improved (so that it offered a lower GOS, or a higher average SIR, or both)?

Note that there is not a single right answer. Depending on the order in which you allocated channels, you may achieve a different GOS and average SIR than others.