1. (25) In the figure below draw the boundary of the region of points in $x_1,x_2$ space that satisfy the following constraints:

$$
\begin{align*}
    x_2 &\leq 1 \\
    x_1 &\leq 1 \\
    -x_1 - x_2 &\leq 0
\end{align*}
$$

Now suppose we want to find values for $x_1$ and $x_2$ that minimize the objective function $x_1 - x_2$ and satisfy the above constraints. In the figure, draw the line representing the objective function so that it passes though the point corresponding to those values. What is the value of the objective function at that point?

On the back of this page, write a Matlab function that finds the optimal values of $x_1$ and $x_2$. 

---

**Instructions:** Answer all questions. Partial credit is considered if you state what you would do with an intermediate result if you were able to derive it.
function ans = prob1()
    f = [1 -1]’;
    A = [0 1 ; 1 0 ; -1 -1];
    b = [1 1 0]’;
    ans = linprog(f,A,b);
end

2. (2) What is the first thing to do when building a class?
   Develop a constructor for the class.

(2) What is “local state” and where does it get defined?
   Local state comprises the variables that are field members of a class. The
   values given to those variables may be different for each object of the
   class. The local state definition and initialization is in the constructor.

(1) What is a method used for?
   A method defines an operation that may be performed on objects of the
   class in which the method is developed.

(20) A stack is a container of objects. Operations on a stack are insert and remove
   which add individual objects to and remove single objects from a stack, respectively.
   However, unlike in the case of a queue, the object removed is the one that was most
   recently inserted. Develop a stack class which supports insert and remove as above.
   Write all Matlab m files that are needed.

function stk = stack()
    stk.lst = {};  
    stk = class(stk,’stack’);  
end

function stk = insert(stk,obj)
    stk.lst{length(stk.lst)+1} = obj;
    assignin(’caller’,inputname(1),stk);
end

function obj = remove(stk)
    if isempty(stk)
        obj = [];
    else
        obj = stk.lst(length(stk.lst));
        stk.lst = stk.lst(1:length(stk.lst)-1);
        assignin(’caller’,inputname(1),stk);
    end
end

function n = isempty(stk)
    if length(stk.lst) == 0 n = 1 ; else n = 0; end
end
You are given some number $n$ of processes. Each process has a deadline measured in seconds (assumed to be a positive integer no greater than $n$), and a profit (measured in U.S. dollars) and these numbers are also known. A processor will work on each process to completion, one at a time, according to a schedule. Each process takes exactly 1 second to complete. If a process is completed on or before its deadline, the person running the processor gets the profit of that process, otherwise the profit for that process is lost. Find the schedule that maximizes the profit for the person running the processor. For example consider the processes shown vertically in the following table:

<table>
<thead>
<tr>
<th>Process #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadline</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Profit</td>
<td>23</td>
<td>19</td>
<td>32</td>
<td>17</td>
<td>21</td>
</tr>
</tbody>
</table>

Then processing in the order 5,1,2,3,4 gives a maximum total profit of 95 where only the profit of process 4 is lost. The problem can be solved several different ways and any good solution will be accepted. First, describe in English (below, on this page) how you would solve the problem for any given set of processes. Second, write something (on the next page) that seems to be a Matlab program using your ideas.

**Algorithm and structure:** Create a vector called `solution`. The value of `solution(j)` will be used to determine which process has been scheduled for time slot $j$. When finished, if `solution(j) > 0`, some process is scheduled for slot $j$. Otherwise, no process is scheduled for that slot. Initially, `solution(j) = 0` for all $1 \leq j \leq n$ which means no processes have been scheduled yet. Sort all processes by decreasing order of profit. Consider each process in turn. If it passes the test below, add it to `solution` as indicated in the test. Otherwise, forget it and move on to the next process.

**The test:** This test procedure will attempt to find a place for a given process $i$ in a partial schedule `solution` and, if a place is found, will put process $i$ in that place. Suppose process $i$ has deadline $d$. Begin at `solution(d)`, then try `solution(d-1)`, then `solution(d-2)`, and so on until `solution(1)`. The first point $j$ at which `solution(j) == 0` is the point at which process $i$ can be inserted into the schedule to beat its deadline without affecting any other scheduled process. So, we set `solution(j) = i`. On the other hand, if `solution(1)` is reached without finding a 0, then process $i$ is not added to the schedule (that is, `solution`).

**Aftermath:** If `solution(j) > 0` then a process has been scheduled successfully (meets its deadline) in the $j^{th}$ time slot.
% File format:
%   <number-of-items>
%   <item-number> <deadline> <profit>
%   ...

function total = dead_mat(filename)
    fid = fopen(filename,'rt');
    if fid < 0
        error('Cannot find the file');
        exit;
    end
    nitems = fscanf(fid,'%d',1); % read number of processes from file
    items = zeros(nitems,3);
    for i=1:nitems items(i,:)=fscanf(fid,'%d',3)'; end
    items = sortrows(items,3);
    solution = zeros(1,nitems);

    % get the results, assume max deadline <= nitems (others can be scheduled)
    total = 0;
    for i=nitems:-1:1 % test procs in decreasing profit order
        j=items(i,2); % deadline for ith proc
        while j >= 1
            if solution(j) == 0 break; end % find a place j in the schedule
            j=j-1; % for it
        end
        if j >= 1 % if there is a free slot j for
            solution(j) = i; % process i then put it there and
            total = total+items(i,3); % update total profit
        end
    end

    % print the results
    fprintf('process time/deadline profit 
');
    fprintf('------- ------------- --------
');
    for j=1:nitems
        i = solution(j);
        if i > 0
            fprintf('%4d:%t\t%d/%d		%d
',items(i,1),j,items(i,2),items(i,3));
        end
    end
    fprintf('Total profit = %d
',total);
end
4. Variables $x_1, x_2, \ldots, x_n$ are random variables that each have probability 1/2 of taking value 1 and probability 1/2 of taking value 0.

(3) What is the mean of $x_1 + x_2 + \ldots + x_n$?

\[ \frac{n}{2} \]

(3) What is the distribution of the sum $x_1 + x_2 + \ldots + x_n$?

binomial: $Pr(x = k) = \binom{n}{k}/2^n$.

(3) Does this look like another important distribution - how do you know?

Yes! It looks like a gaussian distribution - because the sum of a large number of uniformly distributed random variables is itself a random variable which has a gaussian distribution, in the limit, as $n \to \infty$.

(8) Design Matlab code to test this theory. First say in English what the code is supposed to do.

Choose numbers $n$ and $m$. Label $n+1$ binary bins from 0 to $n$. Do the following $m$ times: let $s$ be the ceiling of the sum of $n$ random 0-1 (binary) numbers and add a disk to the binary bin labeled $s$. Label $n+1$ gauss bins from 0 to $n$. Calculate $\sigma = \sqrt{n}/2$ and $\mu = n/2$. Do the following $m$ times: let $s$ be the ceiling of a gaussian random number with parameters $\mu$ and $\sigma$ and add a disk to the gauss bin labeled $s$. Sum the absolute differences of same-label binary and gauss bins, then divide by $m$. The result is the percent difference. This number should go down if: $n$ and $m$ get larger with the same ration; if $m$ gets larger, $n$ fixed.

(8) Next write the code to do it

% m = number of experiments to run, n = number of variables to sum
% stddev(sum) = sqrt(n*((1-.5)^2*(1/2)+(0-.5)^2*(1/2))) = sqrt(n)/2
function prob4(n,m)
    std = sqrt(n)/2;
    mean = n/2;
    binary = zeros(1,n+1);
    gauss = zeros(1,n+1);
    for i=1:m
        s = sum(floor(2*rand(1,n))); % sum of binary
        binary(s+1) = binary(s+1) + 1;
        s = ceil(randn*std + mean + 0.5); % gaussian
        if s >= 1 & s <= n+1 gauss(s) = gauss(s) + 1; end
    end
    % Compare all buckets
    fprintf('Percent misplaced = %4.2f\n',100*sum(abs(binary-gauss))/m);
end