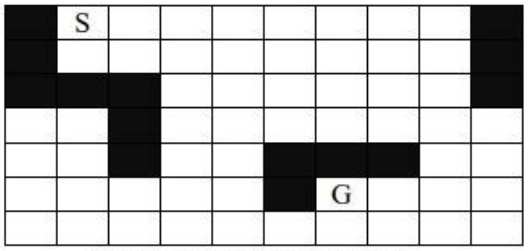
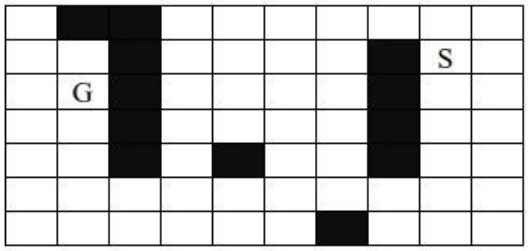
Given the map below (select the one based on your matriculation number), implement a software program for the motion planning method based on the numerical navigation function. The black cells are obstacles, the cell with S is the starting one, the goal cell is denoted by G. If the algorithm is successful, provide the sequence of cells from the start to the goal. It is up to you the choice of the adjacent cells (4, 8, …): brief justify your choice and comment on the result.



Odd matriculation numbers



Even matriculation numbers

For this exercise, the number of adjacent cells chosen is 4. With a number of adjacent cells equal to 8, the algorithm follows a shorter path, but the implementation of the algorithm is computationally more expensive. For the purpose of the exercise, since there are not many cells, 4 has been chosen as the number of adjacent cells. Furthermore, with a number of adjacency cells equal to 8, a non-point-sized robot is more constrained by the dimensions, because moving in an oblique direction it is more inclined to collide with obstacles.

Given the following matrix:

```
clc
clear all
syms B W
%W represents a cell where it is possible to move.
%B represents an obstacle
%0 is the goal point
B=-1

A=[W,B,B,W,W,W,W,W,W,W;
   W,W,B,W,W,W,W,B,W,W;
   W,0,B,W,W,W,W,B,W,W;
   W,W,B,W,W,W,W,B,W,W;
   W,W,B,W,B,W,W,B,W,W;
   W,W,W,W,W,W,W,W,W,W;
   W,W,W,W,W,W,B,W,W,W;]
```

The first step is to appropriately enumerate the reachable cells of the matrix. The value of each cell is checked, and if the cell value is k, a value equal to k + 1 is associated with the adjacent cells that are not obstacles.

```
%while loop ends when all the cells have been numbered.
%If z==k means that no new cells have been numbered, because all of them have already been
while (found==0)
    for i=1:1:7
        for j=1:1:10
            if (A(i,j)==k & i<7 & A(i+1,j)==W)
                A(i+1,j)=k+1;
                z=k+1;
            end
            if (A(i,j)==k & i>1 & A(i-1,j)==W)
                A(i-1,j)=k+1;
                z=k+1;
            end
            if (A(i,j)==k & j<10 & A(i,j+1)==W)
                A(i,j+1)=k+1;
                z=k+1;
            end
            if (A(i,j)==k & j>1 & A(i,j-1)==W)
                A(i,j-1)=k+1;
                z=k+1;
            end

        end
    end

    if(z==k)
        found=true
    end
    k=k+1;
end

fprintf('The results of the navigation function are \n')
A
```

```
The results of the navigation function are

A =

[ 3, -1, -1, 10, 11, 12, 13, 14, 15, 16]
[ 2,  1, -1,  9, 10, 11, 12, -1, 14, 15]
[ 1,  0, -1,  8,  9, 10, 11, -1, 13, 14]
[ 2,  1, -1,  7,  8,  9, 10, -1, 12, 13]
[ 3,  2, -1,  6, -1,  8,  9, -1, 11, 12]
[ 4,  3,  4,  5,  6,  7,  8,  9, 10, 11]
[ 5,  4,  5,  6,  7,  8, -1, 10, 11, 12]
```

```
%With the final part of this algorithm, we proceed backwards, going from one node to
%another node labeled with a lower value, avoid passing through the
%obstacles (-1). The algorithm ends when the goal is reached.
while reached==0
   if(y(t)<10 && ((A(x(t),y(t)+1))<A(x(t),y(t))) && A(x(t),y(t)+1)>=0)
       x(t+1)=x(t);
       y(t+1)=y(t)+1;
   end

   if(x(t)<7 && ((A(x(t)+1,y(t)))<A(x(t),y(t)))  && A(x(t)+1,y(t))>=0)
       x(t+1)=x(t)+1;
       y(t+1)=y(t);
   end

   if(y(t)>1 && ((A(x(t),y(t)-1))<A(x(t),y(t)))  && A(x(t),y(t)-1)>=0)
       x(t+1)=x(t);
       y(t+1)=y(t)-1;
   end

    if(x(t)>1 && ((A(x(t)-1,y(t)))<A(x(t),y(t))) && A(x(t)-1,y(t))>=0)
       x(t+1)=x(t)-1;
       y(t+1)=y(t);
   end
  if(A(x(t),y(t))==0)
       reached=1
  end

  t=t+1;

end
```

the sequence of cells that connects the start with the goal is:

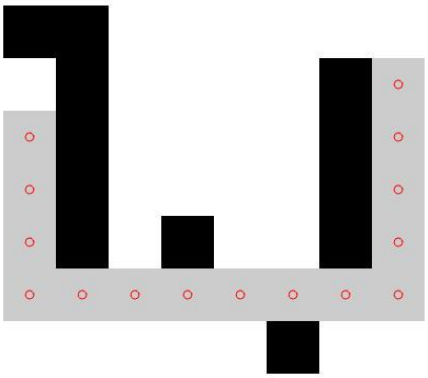
```
x =

     2     3     4     5     6     6     6     6     6     6     6     6     5     4     3


y =

     9     9     9     9     9     8     7     6     5     4     3     2     2     2     2
```

A graphical representation is shown below



The file containing the algorithm is called: "numerical navigation function.m"