QUESTION 1 ***Shortest Paths*** **(6+1 = 7 points)**

Apply the Dijkstra Algorithm for finding the shortest distances from S to A..G

<u>Distances</u>

SA = 3.2	SB = 5.1	SC = 3.6
CG = 7.8	EG = 5.7	DG = 5.1
AB = 2.0	BD = 2.8	DE = 3.2
EC = 2.2	CA = 2.2	

Ready Set = ____________________________

Node	**S**tart	**A**	**B**	**C**	**D**	**E**	**G**oal
Distance	0						
Predecessor	–						

Shortest Distance S-->G ____

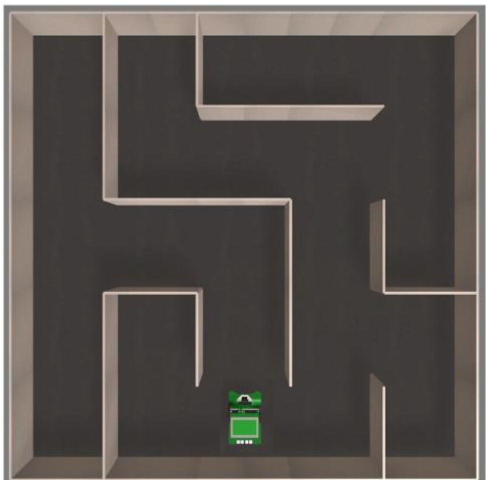
Shortest Path S--> G: ____________________

QUESTION 2 *Maze Navigation* (4 points)

Consider this iterative algorithm:

```
 while(1)
 { if (right_open) VWTurn(-90,100) VWWait();      // turn right
    else if (front_open) ;                        // no turn
     else if (left_open) VWTurn(90,100) VWWait(); // turn left
      else VWTurn(180,100) VWWait();              // turn 180
   VWStraight(100,100); VWWait();}                // go 1 cell
```

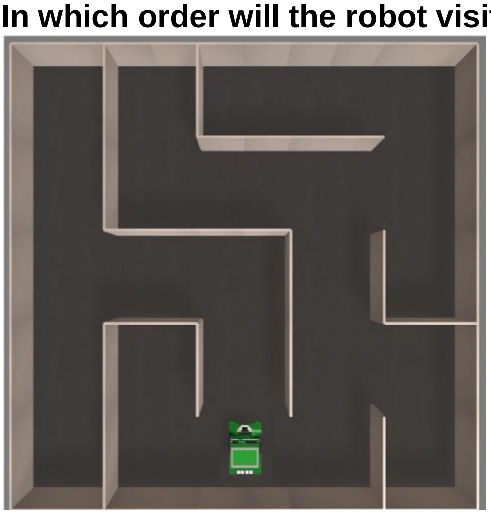
In which order will the robot visit the maze cells?



Consider this recursive algorithm:

```
void explore()
{ "mark current cell";
{ if (front_open  && "front cell unmarked")
    { "drive 1 cell forward";  explore(); "drive 1 cell back";  }
 if (left_open && "left cell unmarked")
    { "drive 1 cell left"; explore(); "drive 1 cell right"; }
 if (right_open && "right cell unmarked")
    { "drive 1 cell right" explore(); "drive 1 cell left";  }
}
```

In which order will the robot visit the maze cells?



QUESTION 3 *Waypoint Navigation* (10 points)

Write a C-function drive_waypoint that drives a mobile robot from its current position to the next waypoint, with absolute position: wx, wy.

- You can assume that there will be a collision-free path to the waypoint.
- Continually read the robot's current pose using functions **VWGetPosition(&x,&y,&phi)**.

```
void drive_waypoint(float wx, float wy)
{
```

QUESTION 4 *Genetic Algorithms* **(6 points)**

Write Pseudocode to describe a Genetic Algorithm, to evolve a robot movement.

- Assume the robot movement can be coded by 50 integers: **int move[50]**
- Use a population of 1,000 genes: **int genepool[1000][50]**
- Use an array of fitness values: **int fitness[1000]**
- Assume an evaluation function exits to assign a fitness value to each movement:
  int eval(int robmove[])
- Assume you have functions available.
  void crossover(int parent1[], int parent2[], int child1[], int child2[]);
  void mutation(robmove[]);
  int select(fit[]);

QUESTION 5 *Differential-Kinematics* (2+2 = 4 points)

For a differential drive vehicle with:

- Wheel radius 4cm
- Left/right wheels are 15cm apart

(a) Forward
Calculate v and ω for

θ'_L = 2.2 rev/s
θ'_R = 1.5 rev/s

v = ___________ m/s ω = ___________ rad/s

(b) Inverse
Calculate θ_L and θ_R for:

- v = 1.5 m/s
- ω = –45 deg/s *(careful with units!)*

θ'_L = ___________ rev/s θ'_R = ___________ rev/s

QUESTION 6 *Ackermann-Kinematics* (2+2 = 4 points)

For an Ackermann-steering vehicle:

- Assume wheel radius is 4cm
- Front/back wheels are 20cm apart

(a) Forward
Calculate the resulting angular velocity.

- v = 1 m/s
- Steering angle 10°

ω = __________________________ rad/s

(b) Inverse
What steering angle is required for a desired angular velocity.

- ω = 2 rad/s
- v = 2 m/s

α = __________________ deg

QUESTION 7 *Omni-Kinematics* (2+2+2+2 = 8 points)

For an omni-directional vehicle with 4 driven Mecanum wheels:

- Assume wheel radius is 4cm
- Left/right wheels are 15cm apart
- Front/back wheels are 20cm apart

Forward
What is the resulting speed for individual wheel speeds:

(a) Θ'_{FL} = 360°/s Θ'_{FR} = –360°/s Θ'_{BL} = –360°/s Θ'_{BR} = 360°/s

v_x= v_y = ω =

(b) Θ'_{FL} = 180°/s Θ'_{FR} = 180°/s Θ'_{BL} = 180°/s Θ'_{BR} = 180°/s

v_x= v_y = ω =

Inverse
Calculate the four individual wheel speeds in °/s for desired vehicle speeds:

(c) v_x= 1m/s, v_y = 1m/s, ω = 0

Θ'_{FL} = Θ'_{FR} = Θ'_{BL} = Θ'_{BR} =

(d) v_x= 2m/s, v_y = 0m/s, ω = 30°/s

Θ'_{FL} = Θ'_{FR} = Θ'_{BL} = Θ'_{BR} =
