Rubrics for mini Project

Students are required to upload a report having contents given below:

1) Detailed Problem statement
2) Functionalities /features supported by project
3) Any special algorithm or method used for implementation
4) Code and output of the program

Total Marks: 10

Sr. No	Performance Indicator	Excellent	Good	Satisfactory	Unsatisfactory
1)	Completeness and correctness [4]	Project is complete. All necessary functionalities are supported. works correctly for all input cases [4M]	Project is complete. Most of the necessary functionalities are supported. Missing few minor details. works correctly for all input cases [3M]	Project is complete. Only major functionalities are supported. Missing minor details. Program functions correctly for most of the input cases.[2M]	Project is incomplete. Supports very few functionalities, Missing many details. Program functions correctly for few input cases.[1M]
2)	Team work [2]	Each member of the team contributed equally and worked efficiently[2]	Each member of the team contributed almost equally and worked efficiently[1.5]	Only two members of the team contributed [1]	team lacks contribution by each member [0.5]
3)	Report [3]	Content is very informative and accurate. It is easy to read and understand. [3M]	Content is informative and mostly accurate. It is readable.[2M]	Content is not always related to topic. It is less readable due to poor sequencing and sentence structure. [1M]	Content is not relevant or accurate. It is unreadable and difficult to understand due to illogical sequencing and sentence structure. [0.5-0M]
4)	Promptness [1]	The project work submitted on time [1 mark]	The project work is submitted on the next day. [0.5 marks]	The project work is submitted late. [0 marks]	

Maze Solver

Group details

- 8596 – batch A
- 8608 – batch B
- 8625 – batch C

Problem Statement

Write a program to find the shortest path from the top left corner to the bottom right corner using Dijkstra's Algorithm in a randomly generated maze of 50x50

Features/functionalities

- Generates a random maze
- Press "d" to apply Dijkstra's algorithm to the maze
- Press "c" to clear the maze path
- The generation of the path and clearing are both animated
- There is a new button which on Click generates a new maze
- The programming language used is Python
- The display is done by Tkinter
- Use of networkx module of python

Algorithm

The Algorithm used in this project is Dijkstra's Algorithm

Code and Output

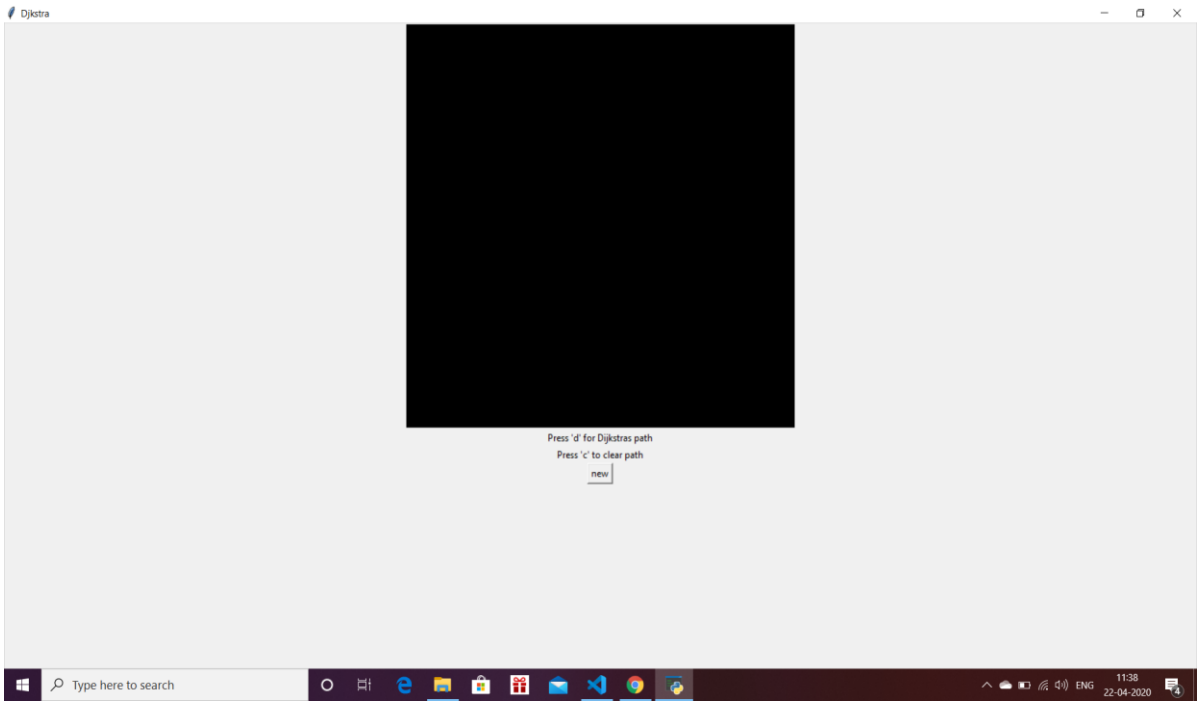
```
import tkinter as tk,random,networkx as nx,time



root = tk.Tk()
root.title("Djkstra")
root.geometry('600x600')
c = tk.Canvas(root, height=500, width=500, bg="black")
c.pack()
lable1 = tk.Label(text = "Press 'd' for Dijkstras path")
```

```
lable2 = tk.Label(text = "Press 'c' to clear path")
lable1.pack(side='top')
lable2.pack(side='top')
dk = []
def new():
    b=c.create_rectangle(0,0,500,500,fill= "#ffafbd")
    x = 10
    while (x != 500):
        c.create_line(0,x,500,x)
        c.create_line(x,0,x,500)
        x=x+10


    matrix =[]
    for i in range (0,50):
        matrix.append([])
    for i in range (0,50):
        for j in range (0,50):
            matrix[i].append(j)
            matrix[i][j]=0
    for j in range (0,50):
        for i in range (0,50):
            matrix[j][i]=random.randint(0,1)*random.randint(0,1)*random.randint(0,1)
            if(matrix[j][i]==1):
                r=c.create_rectangle(i*10,j*10,(i*10)+10,(j*10)+10, fill="#00FFFF")


    matrix[49][49]=0
    matrix[0][0]=0
    g = nx.Graph()
    for i in range (0,50):
        for j in range (0,50):
            if(matrix[i][j]!=1):
```

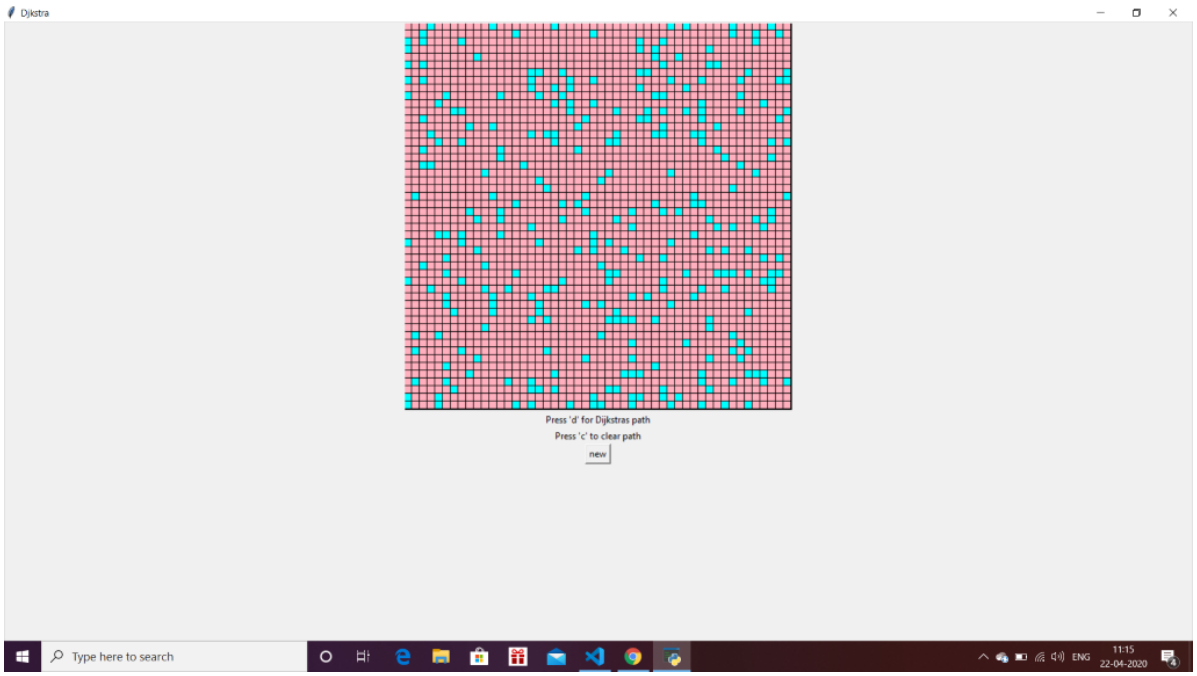
```
                if(j!=49  and matrix[i][j+1]==0 ):
                    g.add_edge((i*50)+j,(i*50)+j+1)
                if(i!=49 and matrix[i+1][j]==0):
                    g.add_edge((i*50)+j,((i+1)*50)+j)


    global dk
    dk =nx.dijkstra_path(g,0,(50*50)-1)

def move(i,j):
    r=c.create_rectangle(i*10,j*10,(i*10)+10,(j*10)+10, fill="black")
    root.update()


def clear(i,j):
    r=c.create_rectangle(i*10,j*10,(i*10)+10,(j*10)+10, fill="#ffafbd")
    root.update()


def function(event):
    global dk
    if(event.char=="d"):
        x = 0
        while(x !=len(dk)):
            b = dk[x]//50
            a = dk[x]%50
            root.after(10,move(a,b))
            x = x+1


    if(event.char=="c"):
        x = 0
        while(x !=len(dk)):
```

```
            b = dk[x]//50
            a = dk[x]%50
            root.after(10,clear(a,b))
            x = x+1

button = tk.Button(text = "new",command = new)
button.pack(side = "top")
r=c.create_rectangle(0,0,10,10, fill="black")
r=c.create_rectangle(490,490,500,500, fill="black")
root.bind("<Key>",function)
root.mainloop()
```

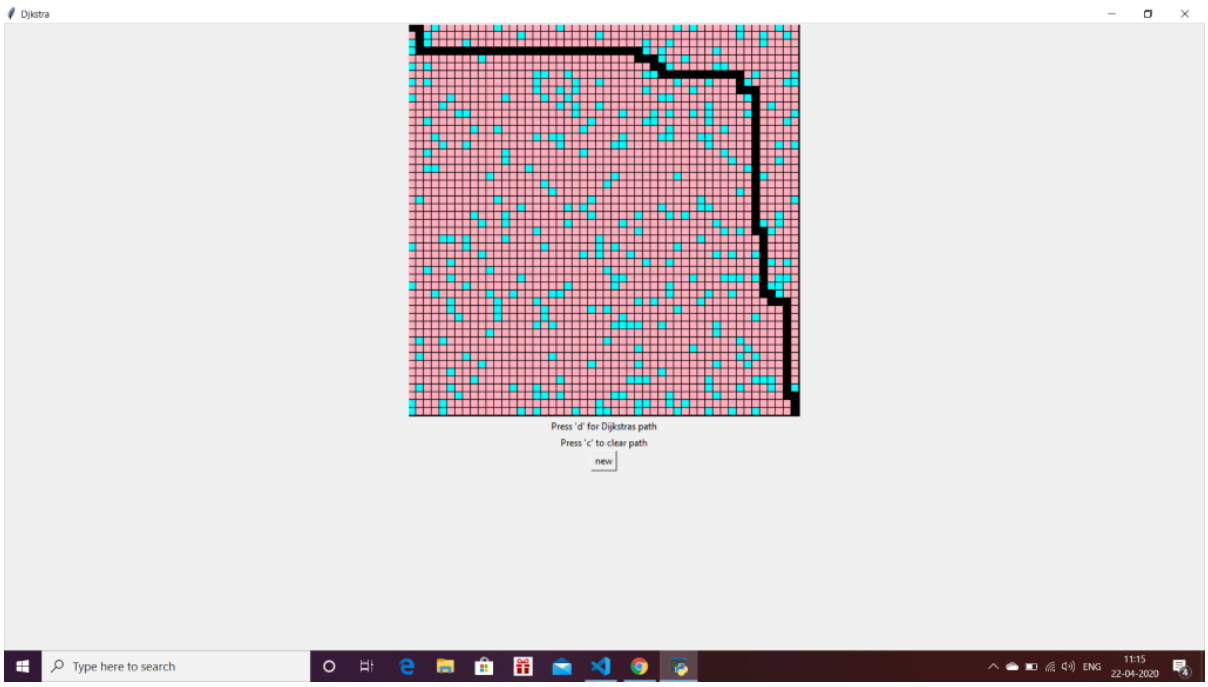
Initial Screen:



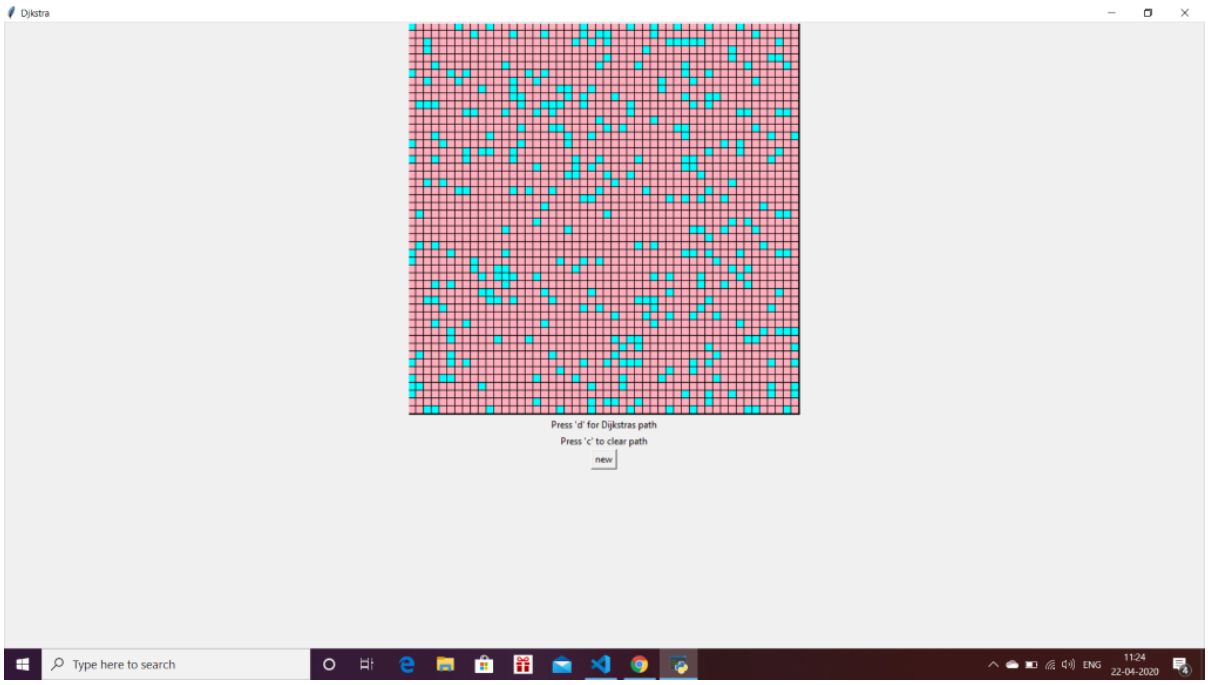
After clicking new:



After pressing ‘d’:



After clicking new again:


Djkstra
Press 'd' for Dijkstras path
Press 'c' to clear path
new
Type here to search
ENG
11:24
22-04-2020