Spring 2020 CSC 113 Intro. To Programming with MATLAB

Final Exam 5/5/2020 Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Complete all programs in MATLAB script files and function files as appropriate.   
You may separate different problems into different files as needed.

1. Write a MATLAB program to create a new figure that is split into three subplots, all in a column, plotting the three graphs listed below: (25 points)
   1. Plot a bar plot showing the point value of each question on this exam (excluding extra credit). Label the x-axis “Question” and the y-axis “Point value.”
   2. Plot the approximate numerical differential of *y* with respect to *x*. Use *x* values of 1, 2, 3, 4, 5 and matching *y* values of 40, 35, 27, 23, 12.
   3. Plot the solution of the following differential equation for values of *t* between 0 and 1, with the initial condition of *y*=0 when *t*=0:

Code

clc

clear

close

% Q01

% Abdulelah Bukdam

subplot 311

bar([1:4],[25,30,15,30])

xlabel('Question')

ylabel('Points')

title('Part a')

x=1:5;

y=[40,35,27,23,12];

diff\_=diff(y)./diff(x);

subplot 312

plot(diff\_)

xlabel('diff(x)')

ylabel('diff(y)/diff(x)')

title('Part b')

tspan = [0 1];

y0 = 0;

[t,y] = ode45(@(t,y) t^3 -y0, tspan, y0);

subplot 313

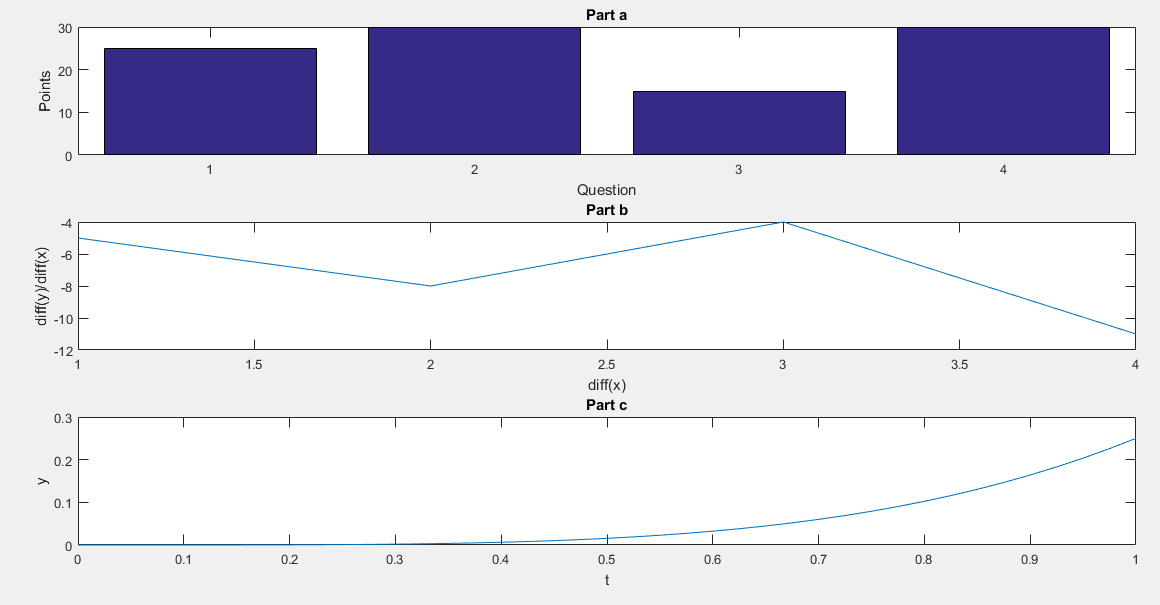
plot(t,y)

xlabel('t')

ylabel('y')

title('Part c')

Output



1. Write a program to display to the screen all the prime numbers between 1 and a user selected number **without** **using MATLAB’s primes(), isPrime() or any other prime-related function** (30 points)**:**
   1. Create a function **listPrimes** that returns a vector of all prime numbers between 1 and any target number given as an input parameter.
   2. Create a script file that asks the user to enter the desired number and uses **listPrimes** to display the requested list of numbers.
   3. Input Validation: Your script should do not accept a negative number from the user. Ask for new input instead.

Code

% Q02

%Abdulelah Bukdam

clc

clear

close

x =input('Enter the value: ');

while (true)

if(x<=0)

x =input('again enter value you enter wrong value : ');

else

break

end

end

primes = 1:x;

p = listPrimes(primes);

primes(p)

function isp = listPrimes(X)

isp = false(size(X));

if ~isempty(X)

X = X(:);

if ~isreal(X) || any(X < 0) || any(floor(X) ~= X) || ...

any(isinf(X))

error(message('value not real'));

end

n = max(X);

if isinteger(X) || n <= flintmax(class(X))

if (isa(X,'uint64') || isa(X,'int64')) && n > flintmax

p = primes(2.^(nextpow2(n)/2));

else

p = primes(cast(sqrt(double(n)),class(X)));

end

for k = 1:numel(isp)

Xk = X(k);

isp(k) = (Xk>1) && all(rem(Xk, p(p<Xk)));

end

else

fm = flintmax(class(X));

p = primes(sqrt(fm));

for k = 1:numel(isp)

Xk = X(k);

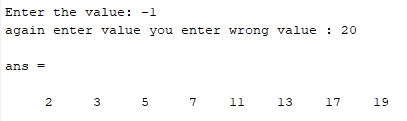
isp(k) = (Xk<fm) && (Xk>1) && all(rem(Xk, p(p<Xk)));

end

end

end

Output



1. Write MATLAB code to solve the following system of linear equations. Display the value of each variable (eg. “*x* = 1, *y* = 2, and *z* = 3”). (15 points)  
   6y - 4𝑧 = 6  
   5𝑥 - 7𝑦 + 3z = -2  
   𝑥 - 2𝑦 = 0

Code

% Q03

%Abdulelah Bukdam

clc

clear

A=[0 6 -4 ;5 -7 3; 1 -2 0];

B=[6;-2;0];

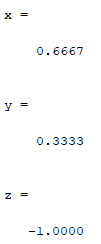
C=inv(A)\*B;

x=C(1)

y=C(2)

z=C(3)

Output



1. Write a MATLAB program that acts as a cost calculator (30 points).
   1. The program should ask the user to enter a price.
   2. It should ask the user to enter an operator: ‘**+’**, ‘-‘, ‘\*’, ‘/’ or ‘=.’
      1. If the user selects =, the program should display the current price in proper dollar and cent format and terminate.
      2. If the user selects any of the other operators, it should ask for a second number and apply the requested mathematical operation to the current price.
   3. The program should continue asking for operators until the user selects ‘**=.’**
   4. Input Validation:
      1. Do not allow the user to enter any operand besides the five listed; ask the user to enter a valid operand instead.
      2. Do not allow division by 0; ask the user to enter a valid divisor instead.

Code

% Q04

%Abdulelah Bukdam

first=input('enter first: ');

op=input('enter op: ','s');

while(1)

while(1)

switch(op)

case '+'

sec\_number=input('enter sec\_number number:');

disp(['Result is :',num2str(first+sec\_number)])

case '-'

sec\_number=input('enter sec\_number number:');

disp(['Result is :',num2str(first-sec\_number)])

case '\*'

sec\_number=input('enter sec\_number number:');

disp(['Result is :',num2str(first\*sec\_number)])

case '/'

sec\_number=input('enter sec\_number number:');

while (1)

if(sec\_number==0)

sec\_number=input('Re enter sec\_number number:');

else

break

end

end

disp(['Result is :',num2str(first/sec\_number)])

case '='

break

otherwise

disp('you choose wrong op')

op=input('Re enter op: ','s');

end

end

if(op=='=')

break

else

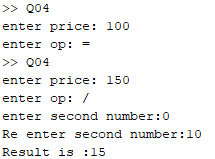
first=input('enter first: ');

op=input('enter op: ','s');

end

end

Output



**Extra Credit:** Write a MATLAB program that simulates the projectile motion of a cannon ball on Earth. Ask the user to enter the launch angle and the initial velocity of the cannon ball. Plot the trajectory (path) the ball travels until it returns to the ground. You will have to research the relevant equations. For simplicity, you may assume the ball is traveling over a flat surface and ignore air resistance. Write a summary of the simulation to a file named “SimData.txt,” including the launch angle, velocity, maximum height achieved, and horizontal distance traveled (include labels and units). Once complete, the user may choose if they want to run another simulation, or exit. (20 points)

Code

% Extra credit

%Abdulelah Bukdam

angle=input('enter angle:');

while (1)

m=94;

A=pi\*(.21)^2;

C=.12;

p0= 1.225;

g=9.81;

%Initial Conditions

dt= .001;

x=0;

y=0;

v=10000;

vx=v\*cosd(angle);

vy=v\*sind(angle);

t=0 ;

y0 = 1000;

i=1;

% Half step

p = p0\*exp(-y/y0);

D = C\*(p\*v^2)/2\*A;

ax = -D/m;

ay = -g;

while y(i) >= 0

i = i+1;

vx(i) = vx(i-1)+ax(i-1)\*dt;

vy(i) = vy(i-1)+ay(i-1)\*dt;

x(i) = x(i-1)+(vx(i-1)+vx(i))/2\*dt;

y(i) = y(i-1)+(vy(i-1)+vy(i))/2\*dt;

p = p0\*exp(-y(i)/y0);

v = sqrt(vx(i)^2+vy(i)^2);

D = C\*(p\*v^2)/2\*A;

ax(i) = -D/m;

ay(i) = -g;

end

figure;

plot(x,y,'-')

xlabel('x distance (m)')

ylabel('y distance (m)')

reuse=input('want to reuse than press y:')

if(reuse=='y')

else

break

end

end

Output

