Maple Lab Test Solutions

Oliver Tan

MATH123/41 - 2013s2

This is based off the all inclusive maple lab test questions (the one with 40 marks).

Notes and Tips

- `**x**` means to plug `x` in place of it
- This resource is not responsible for you losing marks. Do not blame me if you're wrong. I will smite you.
- If in doubt, use `?command`, and `?inifcns` has some good stuff.

You are advised to use these commands are the start of the test:

```
with(LinearAlgebra);
with(plots3d);
```

1

Copy paste the values in, and replace every 5 commas with '`>|<`'

```
Eigenvalues(<<**replaced string**>>);
```

Remember to write your solution in the form provided.

2

Copy paste the values in, and replace every 5 commas with '`>|<`'

```
Rank(<<**replaced string**>>);
```

3

Question is usually in the form

$$\sum_{k=a}^{b}(eqn)^k$$

So the solution is:

```
sum((**eqn**)^k, k=a..b)
```

4

Question is usually in the form

$$\prod_{k=a}^{b} eqn$$

.

```
product(**eqn**, k=a..b)
```

5

```
CrossProduct(**vector 1**, **vector 2**)
```

where vector is of the form `<a,b,c>`.

6

```
**copy paste equations for p and q**
convert(p/q, parfrac);
```

It will be easier to copy paste the answer directly into the Maple Text Box.

7

7.1 a

```
(x,y) -> **eqn**
```

7.2 b

Equation of form

$$\frac{\partial^a}{\partial x^b \partial y^c}$$

```
D[1$**b**, 2$**c**](f)(**xsub**,**ysub**);
```

8

```
dsolve({**eqn**, y(**sub**)=**sub**, D(y)(**sub**)=**sub**},y(x));
```

An example is provided below for ease of use:

$$y\frac{d^2y}{dx^2} + (\frac{dy}{dx})^2 = 0 = 0, y(0) = 1, y'(0) = 9$$

```
diff({y(x)*diff(y(x),x$2) + diff(y(x),x)^2=0, y(0)=1, D(f)(0)=9}, y(x));
```

9

```
dsolve({**eqn**, y(**sub**)=**sub**, D(y)(**sub**)=**sub**},y(x));
```

10

```
dsolve({**eqn**, y(**sub**)=**sub**, D(y)(**sub**)=**sub**},y(x));
```

11

```
dsolve({**eqn**, y(**sub**)=**sub**, D(y)(**sub**)=**sub**},y(x));
```

12

```
point(A, [1,2,3]);
```

13

```
line(L, [A, B]);
```

14

Remember, for parallel lines or normal points for planes, it is of the form point first, then a vector which is in list form.

```
line(L, [A, [1,2,3]]);
```

15

```
plane(P, [A, B, C]);
```

16

Remember, for parallel lines or normal points for planes, it is of the form point first, then a vector which is in list form.

```
plane(P, [A, [1,2,3]]);
```

17

If a number is entered, radius is always considered second.

```
sphere(S, [A, r]);
```

18

```
sphere(S, [A, B]);
```

19

```
sphere(S, [A, B, C, E]);
```

20

```
intersection(T, S1, S2);
```

21

```
center(A, S);
```

22

```
intersection(L, P1, P2);
```

23

```
intersection(A, L, P);
```

24

Note there are comments, which are indicated by the hashes (#). Note semicolons supress (:) output, whereas semicolons (;) do not.

```
point(A,[**vector**]): # e.g. point(A, [1,2,3]):
point(B,[**vector**]):
point(C,[**vector**]):
line(L1,[A,[**vector**]]): # e.g. line(L1, [A,[1,2,3]):
plane(P,[B,[**vector**]]):
intersection(E,L1,P):
sphere(S,[A,B,C,E]):
centre(F,S):
line(L2,[C,F]):

evalf(FindAngle(L1,P)); #a
detail(F); #b, copy paste vector exactly
distance(A,L2); #c
```

25

Note there are comments, which are indicated by the hashes (#).

```
point(A,[**vector**]): # e.g. point(A, [1,2,3]):
point(B,[**vector**]):
point(C,[**vector**]):
sphere(S1,[A,**radius**]):
sphere(S2,[B,C]):
intersection(T,S1,S2):
center(E,T): # blame american spelling
line(L1,[B,E]):
line(L2,[A,[**vector**]):

detail(F); #a, copy paste vector exactly
evalf(FindAngle(L1,P)); #b
distance(A,L2); #c
```

26

Sample solutions are already provided for part a - just look at your results. Remember to hit shift+enter for functions.

```
f := proc (m)
        local a, i;
        a[**?**] := **?**; # define initial conditions
        a[**?**] := **?**;

        for i from **last ? + 1** to m do
                a[i] := **condition, where i = n - 1 (**;
        end do;
        return a[m];
end proc;

f(**val**);
```

For example, for the fibonacci sequence,

```
f := proc(m)
        local a, i;
        a[1] := 1;
        a[2] := 1;

        for i from 3 to m do
                a[i] := a[i-1] + a[i-2];
        end do;
        return a[m];
end proc;

f(20);
```

27

Sample solutions are already provided for part a - just look at your results. Remember to hit shift+enter for functions.

```
**copy paste procedure**
f(**val**); # finds your value
```