Opgave 1. Det rigtige print er print A. Alle tre programmer definerer det samme array på d1, d2, d3, d4 og d5. Et array er en liste af variabelnavne, der kan loopes over

I A outputtes data to gange i do-løkken. Første gang før de variable er tildelt værdier og dernæst en gang mere, hvor der er tildelt endnu en værdi. De nye variabeltilordninger overskriver ikke de gamle, sådan som programmet er skrevet, så for hver observation kommer der blot færre tal i observationslinierne.

Grunden til at det ikke kan være print C er, at regneudtrykket ikke passer. Hovsa(i)=d4 for anden værdi af do-løkken. Dette giver (i=4-1)*3=9. Det at der står 9 i fjerde række for d4 er den eneste forskel mellem A og C (samt i-værdierne). Desuden har variablen i jo andre værdier.

Grunden til at det ikke er B skyldes at der kun er et outputstatement i do-løkken. B har h er halvt så mange observationer fordi der loopes over 2 værdier med kun 1 outputstatement. Dette giver blot 2 observationer. Det første outputstatement er lige i starten af doløkkens værdi – der er således kun tildelt en i-værdi, og derfor er resten blanke. Anden værdi af første doløkke outputtes som 3 for d2. Dette er nu gemt, og kommer derfor i resten af observationerne, da der ikke kommer ny information om d2. d4 kommer først i spil i fjerde observation – den første og tredje var blank, og den anden var d2. d4 bliver således først defineret i fjerde output (anden outputstatement i anden omgang af doløkken).

Opgave 2.

```
data a;
*Her startes med et datastatement. Der arbejdes i et workdatasæt ved navn a;
input bogstav $ x y;
*Der inputtes her 3 variable, hvoraf den første er en bogstavvariabel på højst 8
tegn, kendetegnet ved $;
cards;
a 1 2
b 3 4
c 5 6
d 7 8
e 9 .
;
*Herefter indlæses værdierne af de 3 variable manuelt vha. cards. ;
run;
*Det er egentlig overflødigt med run i lige præcis denne sammenhæng;

*Her definers en makrovariabel med værdien 3. Den henvises til ved at skrive
&tal;
%let tal=3;
*Der påbegyndes nu en makro der kan kaldes ved mult.
%macro mult;
*Makroen påbegyndes med en doløkke fra 1 til 3. Syntaxen ændres indenfor
makroer, og derfor sættes % foran.
%do i=1 %to 3;
data c&i;
*Doløkken køres tre gange for hver værdi af i. Den arbejder på tre datasæt ved
navn henholdsvis c1, c2 og c3. &i er reference til i-værdien idet i nu er dannet
som en makrovariabel.
set a;
```

```
*Det før indlæste datasæt indlæses som c&i;
z=log(x*y);
*Der defineres en ny variabel som z, der er logaritmen til produktet af x og y,
eller summen af logaritmerne til henholdvis x og y.;
if z=. then z=10000;
*Hvis z mangler i datasættet (f.eks. 5. observationer) sættes z lig 10000;
c=&tal||bogstav;
*variable c sammensættes af to værdier. Det første tegn har altid værdien 3 pga.
makrovariablen først, og den anden del har værdien af bogstavvariablen for den
behandlede observation.;
if z>floor(sqrt(&i*&tal)) then delete;
*Her ryddes op i zvariablen. Hvis værdien er større en regneudtrykket fjernes
den fra datsættet;
run;
%end;
*Dette afslutter doløkken i makroen.;
data d;
*Her påbegyndes et nyt datsæt;
set c1 c2 c3;
*Det sammensættes lodret (append) af datsættene c1, c2 og c3 dannet af den
tidligere doløkke;
run;
%mend;
*Dette afslutter makroen;
%mult;
*Dette kalder makroen;* Datasættene er nu genereret;
proc print data=d;
*Her kaldes print proceduren for datasættet d;
run;
*Her eksekveres proceduren;
```

Opgave 3.

Bemærk at der er en trykfejl i opgaven, hvor der mangler et minus i formlen

$$\sum_{i=1}^{\infty}\frac{(-1)^i}{i}=-\log(2)$$

Dette blev rettet et kvarters tid inde i eksamenstiden ved at jeg råbte det højt i begge lokaler.

```
%let k1=10;
%let k2=100;
%let k3=1000;
%let k4=10000;
data sum;
do h=1 to &k1;
minuslog2=-log(2);
k10+((-1)*1)**h/h;
end;
do i=1 to &k2;
k100+((-1)*1)**i/i;
end;
do j=1 to &k3;
```

```
k1000+(-1)**j/j;
end;
do l=1 to &k4;
k10000+(-1)**l/l;
end;
run;
data sum;
set sum;
drop l j i h;
run;
ods rtf body="P:\data\output.rtf";
proc print data=sum noobs;
run;
ods rtf close;
```

Koden består af 4 doløkker kørt til værdien af 4 forskellige makrovariable. Udtrykket k10+ (udtryk) gør at der adderes op og benyttes altså til at udregne summer.
Det giver følgende output.

minuslog2	k10	k100	k1000	k10000
-0.69315	-0.64563	-0.68817	-0.69265	-0.69310

Det ses at for større k-værdier nærmer summen sig –log(2).

Opgave 4.

Jeg har til formålet oprettet et nyt datasæt til at demonstrere.

```
data nytest;
input x;
cards;
0.5
1
2
3
4
5
6
7
8
9
;
run;
```

Et bibliotek kan oprettes ved følgende.

```
libname maj2012 'P:\materiale';
```

Den øverste statement i proc format angiver, at den skal gemme formatet i det formatkatalog der ligger under mappen maj2012.

```
proc format library=maj2012;
value ny /*Formatet navngives ny.*/
1='Et'
1<-6='Få'
```

```
low-<1='Småt'
6<-high='Stort'
;
run;
```

Her tilordnes variablen x formatet ny. Der oprettes desuden en ny variable nyx til at se de oprindelige værdier.

```
data format;
set nytest;
format x ny.;
nyx=x;
run;
ods rtf body="P:\data\output.rtf";
proc print data=format noobs;run;
ods rtf close;
```

Resultatet ses i følgende output genereret vha. proc print og ods rtf. ODS rtf er ligeledes brugt i resten af opgaven til at hente output ud af SAS.

x	nyx
Småt	0.5
Et	1.0
Få	2.0
Få	3.0
Få	4.0
Få	5.0
Få	6.0
Stort	7.0
Stort	8.0
Stort	9.0

Opgave 5.

Der foretages først sikkerhedskopier af de oprindelige datasæt, her ét eksempel.

```
data frafald2;
set sasprog.frafald;
run;
```

Ved hjælp af følgende kode kan der foretages diskriminansanalyse.

```
proc discrim data=frafald2 outstat=disk crosslist list crossvalidate;
class frafald;
```

run;

Der skal her klassificieres ind i de mulige værdier af frafald (0 og 1). Linien outstat gemmer diskrimansresultatet til brug i en senere test. Options crosslist, list og crossvalidate giver ekstra oplysninger, hvoraf den mest interessante er posteriors, h vor man kan se med hvilken sandsynlighed hver observation blev tilføjet hver gruppe. De første 17 er indsat nedenunder. Der er meget output til proc discrim, og jeg har her blot valgt det mest gængse.

Posterior Probability of Membership in Frafald					
Obs	**From Frafald**	**Classified into Frafald**		**0**	**1**
1	**1**	**1**		0.0000	1.0000
2	**1**	**1**		0.0000	1.0000
3	**1**	**1**		0.1290	0.8710
4	**1**	**1**		0.0000	1.0000
5	**1**	**1**		0.0010	0.9990
6	**1**	**0**	*	0.5441	0.4559
7	**1**	**1**		0.0002	0.9998
8	**1**	**1**		0.0000	1.0000
9	**1**	**1**		0.0244	0.9756
10	**1**	**0**	*	0.8558	0.1442
11	**1**	**1**		0.0239	0.9761
12	**1**	**1**		0.0000	1.0000
13	**1**	**0**	*	0.9541	0.0459
14	**1**	**1**		0.0040	0.9960
15	**1**	**1**		0.0000	1.0000
16	**1**	**0**	*	0.8948	0.1052
17	**1**	**1**		0.0006	0.9994

Det ses generelt at der klassificeres med forholdsvis høj sandsynlighed i hvert enkelt tilfælde.
Den mest interessante del af output er oversigten over hvor mange der blev klassificeret rigtigt.

Number of Observations and Percent Classified into Frafald			
From Frafald	**0**	**1**	**Total**
0	485 97.59	12 2.41	497 100.00
1	22 37.93	36 62.07	58 100.00
Total	507 91.35	48 8.65	555 100.00
Priors	0.5	0.5	

Ud af 497 uden frafald blev 484 klassificeret rigtig, dvs. 97 %. Det gik dog knap så godt med dem der er faldet fra. Her er blot 62 % blevet klassificeret rigtigt. Dette skyldes sandsynligvis at der kun er 58 observationer i denne gruppe – dette kan være for lidt til at opbygge regler om diskriminans.

Opgave 5.2.

De fundne diskriminansregler gemt i outstat skal nu bruges til at teste et datasæt og se, hvilken en af de to kategorier de hører til.
Det gøres med følgende kode.

```
proc discrim data=disk testdata=test testlist;
class frafald;
run;
```

Datasættet vælges som der skal opbygges regel efter. Testdata er det ønskede testede datasæt, og optionen testlist tilføjer blandt andet posteriors. Output ses nedenunder. Igen er posteriors og oversigten hevet frem.

Igen ses her posteriors. Stjernen angiver, at observationen ikke er klassificeret i dens oprindelige kategori. Dette er dog ikke så underligt, da samtlige observationer har missing i variablen frafald, og man derfor ikke kender deres oprindelige værdi.
Her ses oversigten.
Det ses at 2 er faldet fra, mens 4 er på gymnasiet ifølge diskriminansanalysen.

Posterior Probability of Membership in Frafald					
Obs	From frafald	Classified into Frafald		0	1
1	.	0	*	0.9885	0.0115
2	.	1	*	0.0009	0.9991
3	.	0	*	0.9831	0.0169
4	.	0	*	0.7677	0.2323
5	.	0	*	0.9439	0.0561
6	.	1	*	0.0291	0.9709

Number of Observations and Percent Classified into Frafald			
From frafald	0	1	Total
.	4 66.67	2 33.33	6 100.00
Total	4 66.67	2 33.33	6 100.00
Priors	0.5	0.5	

Opgave 6.

Der foretages først sikkerhedskopier af de oprindelige datasæt som før. Her foretages multipel imputation. Vha. de eksisterende værdier i datasættet for de observationer med fuld data estimeres sammenhænge mellem de forskellige variable. Dette gøres på et antal af metoder. nimpute= angiver antallet af imputationsmetoder. Antallet er her 5 (også default). Desuden behøves et random seed, her angivet til et tal forskellig fra nul så resultaterne kan replikeres.

```
proc mi data=miss seed=125 out=miskidt nimpute=5;
run;
```

Først forsøges at imputere uden nogle imputationer. Med 12-trins skalaen kan snittet ikke være mindre end -3, og fravær er en procentsats mellem 0 og 100. Vha. proc means studeres outliers.

```
proc means data=miskidt;
run;
```

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
Imputation	Imputation Number	3745	3.0000000	1.4144024	1.0000000	5.0000000
karakter	karakter	3745	6.2432586	2.4718202	-6.2981170	13.9079334
fravaer	fravaer	3745	10.4570296	11.9804923	-16.5560337	88.8000000
Frafald	Frafald	3745	0.1428571	0.3499738	0	1.0000000

Det ses at der opstår et problem med både karakter og fravaer, der får for små værdier. Desuden springer den højeste karakter skalaen. Der forsøges nu med en log transformation. log transformationen foregår i sas ved at danne variablen log(var+c). Der adderes et tal marginalt tæt på 0 (c-værdien), idet man ikke kan tage logaritmen til 0. Default i log transformationen er c=0.
Der foretages igen proc means for at studere resultatet.

```
proc mi data=miss seed=125 out=mi nimpute=5;
transform log(fravaer/c=0.01);
run;
proc means data=mi;
run;
```

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
Imputation	Imputation Number	3745	3.0000000	1.4144024	1.0000000	5.0000000
karakter	karakter	3745	6.3635155	2.2751586	-3.0000000	13.8889243
fravaer	fravaer	3745	10.5120092	12.0176297	0	88.8000000
Frafald	Frafald	3745	0.1428571	0.3499738	0	1.0000000

Minimum giver nu fornuftige grænser. Det eneste problem her er en fortsat høj maks karakter, men med de forskellige muligheder for at gange sit snit er det måske ikke engang helt umuligt. Det må betragtes som en væsentlig forbedring i forhold til det tidligere resultat.

Opgave 6.2.

Først foretages følgende regression.

```
proc reg data=mi outest=estim covout noprint;
by _imputation_;
model karakter=fravaer frafald;
run;
```

Den foretages for hver imputationsnummer – dvs. her 5 gange. Estimaterne gemmes i et datasæt ved navn estim i workbiblioteket. Covout tilføjer output benyttet i mianalyze og noprint gør at oplysningerne ikke kommer i outputvinduet. Nu er der klar til proc mianalyze.

```
proc mianalyze data=estim theta0=0 0 -1;
modeleffects intercept fravaer frafald;
test fravaer=0, frafald=0 /mult;
run;
```

Proceduren benytter datasættet genereret via proc reg. Theta0= tester parameterværdien mod givne nulhypoteser. Jeg tester her at værdien af intercept er lig 0, koefficienten på fravaer er lig 0 og koefficienten på frafald er lig -1.
Først ses her parameter estimaterne.

Parameter Estimates							
Parameter	**Estimate**	**Std Error**	**95% Confidence Limits**		**DF**	**Minimum**	**Maximum**
intercept	7.096780	0.110880	6.87699	7.31657	107.81	7.052329	7.152201
fravaer	-0.045876	0.009542	-0.06483	-0.02692	92.008	-0.050113	-0.041514
frafald	-1.755614	0.307962	-2.36122	-1.15001	364.77	-1.871291	-1.646502

Her er testet for om hældningskoefficienterne på frafald og fravaer er lig med 0. Det ses at de begge meget signifikante, og de anslås altså at have stor forklaringsgrad til karakter.
Til sidst bruges et F-test til at tjekke modellens forklaringsgrad af de to forklarende variable (vha. optionen /mult i test-statement). Ud fra t-testet er et godt gæt at nulhypotesen afvises.

Multivariate Inference Assuming Proportionality of Between/Within Covariance Matrices				
Avg Relative Increase in Variance	**Num DF**	**Den DF**	**F for H0: Parameter=Theta0**	**Pr > F**
0.328636	2	47.09	80.36	<.0001

Nulhypotesen afvises da også med en meget lille p-værdi.
Som den sidste del af output fra mianalyze kigges på variansen.

Variance Information							
	Variance				**Relative**	**Fraction**	
Parameter	**Between**	**Within**	**Total**	**DF**	**Increase in Variance**	**Missing Information**	**Relative Efficiency**
intercept	0.001973	0.009926	0.012294	107.81	0.238567	0.207187	0.960211
fravaer	0.000015821	0.000072068	0.000091053	92.008	0.263433	0.225167	0.956907
frafald	0.008276	0.084909	0.094841	364.77	0.116966	0.109586	0.978553

Within variansen er den varians der altid er i datasættet og som ikke kan fjernes. Between er den varians der er tilføjet vha. multipel imputation. Som tommelfingerregel må denne ikke være større end withinvariansen. I søjlen ”relative increase in variance” ses between/within. Denne skal være mindre end 1. Det ses at between varians højst er en fjerdedel i forhold til within. Det er altså en rimelig imputation målt på dette parameter.