
Gruppedannelse i gymnasiet

- Gruppedannelse i naturvidenskabelige fag -

P1-rapport
Grp. SW1A317b



Aalborg Universitet
Software



AALBORG UNIVERSITET

STUDENTERRAPPORT

Elektronik og IT
Aalborg Universitet
<http://www.aau.dk>

Titel:

Gruppedannelse i gymnasiet

Tema:

Fra eksisterende software til modeller

Projektperiode:

Efterårssemestret 2018

Projektgruppe:

SW1A317b

Deltager(e):

Christopher Lykke Davids
Johan Alexander Turesson Krogh
Frederik Halkjær Olesen
Morten Uldall Vind Nielsen
Adrian Zippor Plesner
Emil Kristian Rasmussen
Rikke Husted Østergaard

Vejleder(e):

Hovedvejleder: Eike Schneiders
Bivejleder: Sahar Sattari

Oplagstal: 1

Sidetæl: 67

Bilagsantal: 8

Afleveringsdato:

19. december 2018

Abstract:

This paper will analyze and discuss group-making theories. It will make use of already-established theories to give a perspective on how to make a group in gymnasiums with as many satisfied students as possible. The paper focuses on science related subjects for making groups.

We found our problem to be: "how can software improve the group making process in gymnasiums so better results are obtained and conflicts are remedied based on the students' team roles and ambition level?"

We successfully implement a model capable of creating groups in any given educational environment. We base our implementation on some of the theories earlier analyzed and discussed.

In collaboration with other AAU groups we collected data to test the algorithm, and found that the groups are well sorted in correlation with the framework we designed.

Rapportens indhold er frit tilgængeligt, men offentliggørelse (med kildeangivelse) må kun ske efter aftale med forfatterne.

Indhold

Forord	vi
Indledning	1
1 Problemanalyse	2
1.1 Elevernes holdning	2
1.2 Fundamentet for "den gode gruppe"	3
1.2.1 Homogene og heterogene grupper	4
1.2.2 Cooperative Learning	4
1.3 Grupperoller	5
1.3.1 Belbins grupperoller	5
1.4 Personlighedstyper til gruppedannelse	7
1.4.1 Myers-Briggs personlighedstyper	7
1.5 Konflikter i gruppen	10
1.6 Afgrænsning til naturfaglige fag	11
1.7 Spørgeskema	11
1.7.1 Spørgsmål 1	12
1.7.2 Spørgsmål 2	12
1.7.3 Spørgsmål 3	13
1.7.4 Spørgsmål 4	14
1.7.5 Spørgsmål 5	14
1.7.6 Spørgsmål 6	15
1.7.7 Spørgsmål 7	16
1.7.8 Refleksion	17
2 Problemformulering	18
3 Problemløsning	19
3.1 Løsningsforslag	19

3.2	Kravspecifikation	20
3.3	Beskrivelse af software	22
3.3.1	Struct student	23
3.3.2	Tekstfil-input	24
3.3.3	Dannelse af faglige grupper	26
3.3.4	Dannelse af sociale grupper	31
3.3.5	Output	43
3.4	Test af programmet	44
4	Evaluering og konklusion	47
4.1	Diskussion	47
4.2	Konklusion	47
4.3	Forbedring af programmet	48
	Bibliografi	50
A	Samtale med Simon Søndergaard Christensen	52
B	Samtale med Nanna Louise Ben-Or	53
C	Input eksempel	54
D	Komplet kodeudsnit af sortBelbin funktionen	55
E	Komplet kodeudsnit af sortWishes og wishedCmp funktionen	57
F	Inputskabelon	65
G	Dataeksempel	66
H	Nyt P1-projektforslag	67
H.1	Prissammenligner for dagligvarer	67

Forord

Denne projektrapport er udarbejdet på 1. semester under Software uddannelsen på Aalborg Universitet.

Kildemetoden, der bliver anvendt igennem rapporten, er Vancouver metoden. I henhold til studieordningen er der udarbejdet et nyt projektforslag, dette findes i bilag H.

Aalborg University, 19. december 2018



Christopher Lykke Davids
<cdavid18@student.aau.dk>



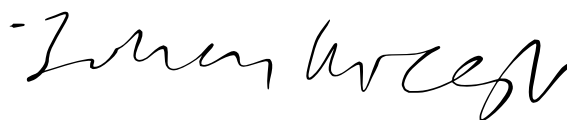
Morten Uldall Vind Nielsen
<muvn18@student.aau.dk>



Adrian Zippor Plesner
<aplesn18@student.aau.dk>



Rikke Husted Østergaard
<raster16@student.aau.dk>



Johan Alexander Turesson Krogh
<jkrogh18@student.aau.dk>



Frederik Halkjær Olesen
<folese18@student.aau.dk>



Emil Kristian Rasmussen
<erasmu18@student.aau.dk>

Indledning

På gymnasiale uddannelser kan det være tidskrævende og problematisk at danne grupper, som fungerer både socialt og fagligt. Det er dog muligt for læreren at minimere nogle af disse problemer med konventionelle gruppedannelsesmetoder. Vælger læreren at danne tilfældige grupper, er der ikke fokus på, hvordan grupperne kommer til at fungere hverken socialt eller fagligt. Danner læreren derimod bevidst konstruerede grupper, kan dette være tidskrævende. Gymnasielærer Anne Boie Johannesson^[1] har bl.a. beskrevet, at de bedste grupper er baseret på, at eleverne kan arbejde sammen. Dette kræver, at de er socialt kompatible med hinanden. Hun har beskrevet, at grupper dannet med udgangspunkt i elevernes faglighed ikke nødvendigvis er velfungerende grupper, da elever med samme faglige niveau ikke nødvendigvis arbejder godt sammen. Dette understøttes af sociolog Simon Søndergaard Christensen, se bilag A, som har forklaret, hvad der ligger bag resultatet af en dårlig og god gruppedannelsesproces. Christensen forklarer de følelser, som gruppemedlemmerne udtrykker, når de er havnet i en gruppe, de er utilfredse med. Dette udtrykkes som afmagt, frustration eller lignende og har rod i gruppemedlemmernes følelse af mangel på indflydelse på gruppedannelsesprocessen. Modsat ses der ved positiv gruppedannelse, at eleverne netop føler, at de er enige med fremgangsmåden. Dette stemmer overens med Johannessons metode, hvor hun spørger eleverne, hvem de gerne vil være sammen med, og de derfor føler, at de har medbestemmelse.

Hvordan kan bevidst konstrueret gruppedannelse garantere et vellykket gruppeprojektarbejde?

Denne rapport vil analysere og diskutere teorier, som vil præsentere relevante perspektiver. Vi vil igennem analyse af teori indskrænke til de parametre, der frembringer et alternativt værktøj, hvor elevernes indflydelse, enten direkte eller indirekte, påvirker programmet.

Vi vil altså have fokus på eleverne, og forsøge at nedsætte de dårlige oplevelser, elever i dag bliver udsat for ved konventionelle gruppedannelsesmetoder.

1. Problemanalyse

Problemanalysen vil fokusere på de forskellige gruppedannelsesmetoder, der anvendes på gymnasier på nuværende tidspunkt. Herefter vil der blive set på homogene og heterogene grupper samt deres effekt på gruppearbejde, efterfulgt af personkarakteriserende teorier som Belbins grupperoller og Myers-Briggs personlighedstyper. Desuden vil de forskellige konflikter, som kan opstå i en gruppe, blive forklaret. Dette vil bidrage til en undersøgelse af hvilken dynamik, der resulterer i enkelte gruppemedlemmers følelse af afmagt, og dertil fremme et forslag til en gruppedannelse, hvor samtlige elever oplever tilfredshed i den nydannede gruppe. Til sidst bliver spørgeskemaresultaterne fra adspurgte elever på de gymnasiale uddannelser gennemgået.

Dette afsnit vil afgrænse definitionen af de tre hyppigst anvendte gruppedannelsesmetoder i gymnasiet.^[2]

- Elevvalgte
- Tilfældighedsbestemte
- Bevidst konstruerede

Disse gruppedannelsesmetoder har hver deres styrker og svagheder, som vil blive analyseret og beskrevet i det efterfølgende afsnit.

1.1 Gymnasieelevers holdninger til nuværende gruppedannelsesmetoder

Når gymnasiet er en fremmed verden^[3] er en bog udarbejdet af Lars Ulriksen et. al., der blev skrevet ud fra undersøgelser gennemført fra 2005 til 2008. Bogen forsøger at forklare hvorfor elever, hvis forældre ikke har en gymnasial uddannelse, har større risiko for at afbryde deres egen uddannelse, og hvorvidt disse elever generelt får lavere karaktergennemsnit end elever, hvis forældre har en gymnasieuddannelse.

Forfatterne har i fælleskab opstillet en række interviews med elever fra de fire gymnasiale ungdomsuddannelser: STX (det almene gymnasium), HF (højere forberedelseseksamen), HHX (højere handelseksamen) og HTX (højere teknisk eksamen). I bogen kommer de bl.a. ind på gruppeprocesser og gruppedannelse i gymnasiet. I disse interviews stilles der spørgsmål om

de umiddelbare fordele og ulemper, der opstår ved nuværende metoder til gruppedannelse. På de gymnasier, hvor eleverne blev interviewet, bestod det hovedsageligt enten af lærervalgt gruppedannelse opnået gennem tilfældighed eller små forsøg på bedre gruppedannelse; ellers stod det til eleverne selv at danne grupper. Eleverne får i disse interviews muligheden for at udtrykke deres tanker og erfaringer om de positive og negative udfald, der forekommer under gruppedannelse.

Eleverne pointerer med hensyn til selvvalgte grupper, at der findes en sikkerhed i selv at herske over, hvem man danner gruppe med, på samme vis som Christensen, se bilag [A](#), definerer den positive gruppedannelse. Eleverne siger, der ligger en tryghed i at have kendskab til gruppemedlemmerne fra starten af. Tidligere erfaring med gruppemedlemmer kan styrke gruppearbejdet, således kan der være mere fokus på selve opgaven frem for den sociale struktur i gruppen. Eleverne påpeger dog, at selvvalgte grupper kan udstøde nogle i klassen. Dertil er faste grupper ligeledes en fordelagtig metode til gruppedannelse, grundet den udvikling man kan erfare i samspil med de samme personer til hvert projekt. Eleverne argumenterer for, at svagheden ved faste grupper er, at grupperne er med til at forme klikker i klassen. En løsning kunne være skiftende grupper, der bryder førnævnte klikker og former et større fælleskab.

I forhold til de dannede grupper, svarer eleverne, at hvis samme niveau - enten fagligt eller socialt - er gældende i gruppen, er det lettere at nå et fælles mål. I en gruppe med svagere stillede elever er det en fordel at have dygtige elever med i gruppen til at hjælpe og derved hæve udbyttet. Lærervalgte grupper kan betyde, at kemien i gruppen ikke fungerer, men hvis gruppedannelsen er en succes, kan det øge muligheden for, at alle lærer noget, samt at de dårligste kommer med i fællesskabet. Derudover har læreren mulighed for at sortere dovne og useriøse elever fra deres mere flittige og engagerede klassekammerater.

Det er altså svært at afgøre, helt præcist hvilken af de konventionelle gruppedannelsesmetoder, elever mener, er den bedste. Grunden til dette er bl.a., at hvorvidt en gruppe er god eller ej afhænger af flere faktorer, herunder det ønskede udbytte, hvilket fag det drejer sig om, om det er i forhold til lærerens eller elevernes synspunkt, og yderligere om dette er de fagligt svage eller stærke elevers synspunkt.

1.2 Fundamentet for "den gode gruppe"

Dette afsnit vil prøve at definere, hvad "den gode gruppe" er, og sætte fokus på bevidst konstruerede grupper. Dette gøres, da denne metode giver læreren bedre kontrol over, hvordan grupperne bliver dannet i modsætning til tilfældigt sammensatte grupper og elevvalgte grupper. Læreren kan vælge at inkludere elevernes egne ønsker, hvis de finder dette fordelagtigt. Ulempen ved denne metode er dog, at den kræver et langt bedre kendskab til elevernes indi-

viduelle karakteristika, hvilket kan være tidskrævende, specielt når målet er at danne grupper med det størst mulige faglige eller sociale udbytte. Til gengæld er metodens største ulempe - mængden af tid det tager - også noget, som man kan have stor indflydelse på gennem en softwarebaseret løsning. Ergo virker det til, at denne ulempe opvejes af metodens vigtigste fordel: at man kan tage højde for den enkelte elevs foretrukne læringsstil, karakterer, personlighed, indflydelsesønsker og lignende samt skræddersy faktorer, så de passer til forskellige grupper og omstændigheder. [2]

1.2.1 Homogene og heterogene grupper

Bevidst konstruerede grupper kan yderligere opdeles i to underkategorier: enten homogene eller heterogene grupper.

I homogene grupper grupperes individer således, at alle befinder sig i en kreds, hvis karakteristika er omtrent lig med deres egne, f.eks. hvis en skole placerede elever i grupper, hvor alle med samme karakter var i samme gruppe. [4]

Heterogene grupper tager også udgangspunkt i sammenhængen mellem individernes karakteristika, dog er fremgangsmåden modsat; i stedet for at samle ens karakteristika i grupper for sig, spredes disse ud så vidt muligt, så der opnås større variation internt i gruppen.

I læringsorienterede sammenhænge ses ofte et større brug af heterogen gruppedannelse frem for dens homogene modsætning. Dette skyldes bl.a., at grupper bestående af elever med forskellige færdigheder i mange tilfælde vil få et større udbytte af gruppearbejdet. Elever vil kunne lære de færdigheder fra sig, som andre mangler, og på samme tid selv udvide deres egen forståelse af emnet gennem denne peer-to-peer-undervisning - "learning by teaching." [2][4]

1.2.2 Cooperative Learning

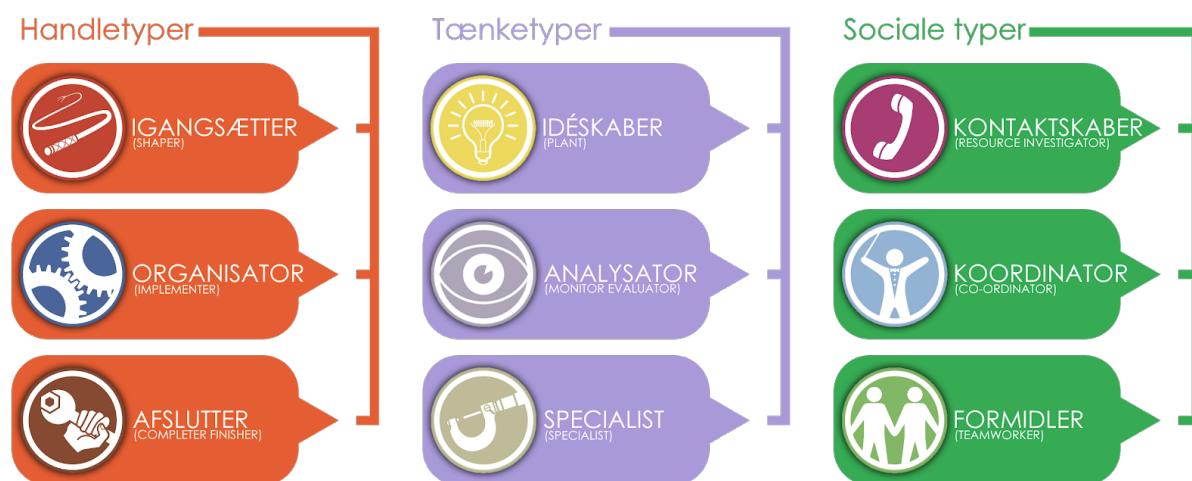
Cooperative learning tager brug af den heterogene tilgang til gruppedannelse. Elever bliver udvalgt baseret på deres faglige resultater og danner grupper således. I formelle cooperative learning-grupper har læreren nogle forskellige roller, som skal opfyldes. Den vigtigste af disse er at tage beslutningerne. Læreren skal både bedømme den enkelte elevs akademiske og sociale færdigheder, hvorefter det er op til læreren at sætte grupperne sammen. Med cooperative learning skal grupperne helst være heterogene, f.eks. i grupper af fire kunne gruppen sættes sammen af en fagligt svagere elev, en fagligt stærkere elev og yderligere to elever, der fagligt ligger i midten. [5]

1.3 Grupperoller

Når man arbejder sammen i en gruppe, vil medlemmerne i gruppen helt af sig selv indtage forskellige roller. Alle har en bestemt rolle, som de foretrækker at påtage sig, når der arbejdes i en gruppe. Den optimale gruppe kan defineres som én, der indeholder en god blanding af roller, således at alle områder er dækkede. Derfor er det også væsentligt, allerede ved gruppedannelsen at sammensætte en gruppe bestående af personer, der indtager forskellige roller.^[6]

1.3.1 Belbins grupperoller

På baggrund af et 9-årigt studium af over 120 grupper har Dr. Meredith Belbin^[1] identificeret 9 forskellige grupperoller på baggrund af den generelle adfærd, han kunne observere blandt medlemmer i grupperne.



Figur 1.1: Belbins grupperoller

Han observerede, at balance i grupperollefordelingen blandt medlemmerne var det vigtigste for dannelsen af en god gruppe. Ved en sådan balance i fordelingen opvejes den enkeltes negative sider, af de andres positive sider. Hvert individ dækker - i forskellige grader - flere roller indenfor Belbins teori, hvilket betyder, at ét individ påtager sig disse roller, når det kommer til gruppearbejde. Derfor er det ikke nødvendigt at have ni gruppemedlemmer for at dække alle rollerne.^{[8][9]}

¹Dr. Belbin er verdenskendt for sit arbejde med at studere grupper og deres individers opførsel. Hans teori om de 9 grupperoller blev formuleret efter et 9-årigt studium af grupper og deres arbejde. Denne teori anvendes succesfuldt i dag verden over, både i undervisning samt på arbejdspladser.^[7]

Belbins grupperoller er inddelt i tre hovedtyper: Handletyper, tænker typer og sociale typer, som vist på figur 1.1.

Handletyper:

Handletyperne indeholder igangsætter, organisator og afslutter. Disse grupperoller er kendetegnet ved, at de får ting gjort, sørger for at holde gruppen i gang, og planlægger arbejdet. De er vigtige for at gruppearbejdet får struktur og en god sammenhæng.

Tænker typer:

Tænker typerne dækker over idéskaber, analysator og specialist. Disse er de kreative og tænkende grupperoller, der tænker ud af boksen og kommer med idéer. De går i dybden med detaljer, når det omhandler de vigtigste ting for et projekt. De er vigtige for, at gruppearbejdet kommer godt i gang og holder et fokuseret spor.

Sociale typer:

De sociale typer er kontaktskaber, koordinater og formidler. Disse grupperoller sørger for at holde sammen på gruppen og løser konflikter. Desuden sørger de for, at gruppearbejdet overholder det overordnede formål, samt uddelegering af arbejdet. De er vigtige for, at gruppearbejdet holder det samme mål, og for at gruppen kan arbejde sammen og kommunikere.

Dette er dog blot én måde at inddele individer på - her med fokus på velfungerende gruppearbejde. En anden måde er f.eks. Myers-Briggs personlighedstypeteori.

1.4 Personlighedstyper til gruppedannelse

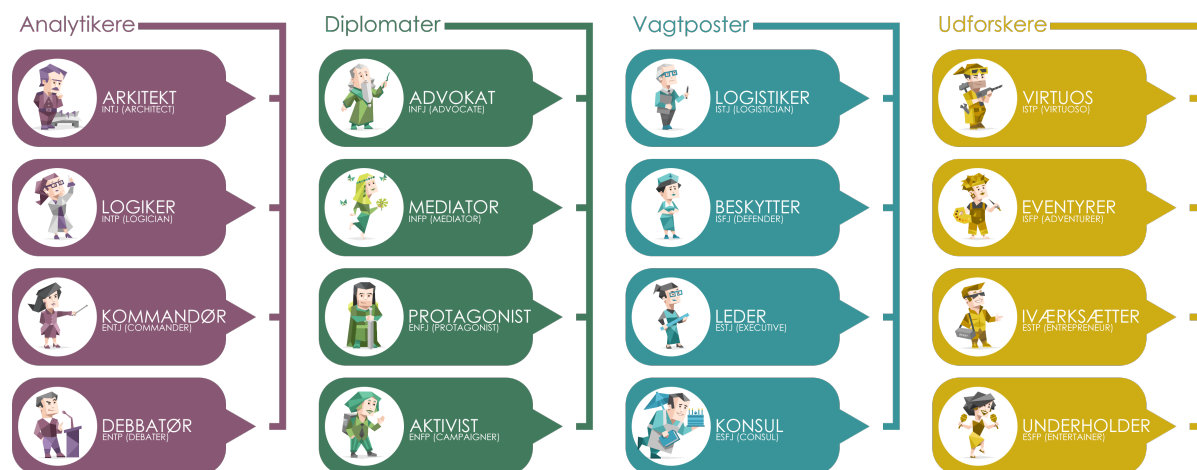
Det kan, ligesom med de forskellige grupperoller, være vigtigt at have en blanding af forskellige personlighedstyper. En af de mest anvendte metoder for angivelse af personlighedstyper er Myers-Briggs personlighedsteori, også kendt som MBTI. MBTI blev udviklet af Katharine Cook Briggs og Isabel Briggs Myers.^[10]

1.4.1 Myers-Briggs personlighedstyper

MBTI baserer personligheder på fire forskellige psykologiske adfærdsområder - sind (mind), opfattelse (energy), natur (nature) og taktik (tactics), samt en ekstra: Identitet (identity) - der hver afgøres af personens største tilbøjelighed på en skala mellem to modsatte adfærdsmønstre. Dette skaber i alt 16 forskellige personlighedstyper. MBTI-testen består af udsagn, hvortil brugeren angiver graden af deres enighed eller uenighed. De forskellige personlighedstyper består hver af en unik bogstavkombination, med hvert bogstav baseret på et af de fire bestemte psykologiske områder, som forklarer personens psykologiske adfærd.

Ifølge testen kategoriseres testpersonen enten som Introvert (I) eller Ekstrovert (E) af sind, personens opfattelse som baseret på Intuition (N) eller Sansning (S), personens planlægning og beslutningstagen som baseret på enten Tanker (T) eller Følelser (F), og til sidst hvorvidt testpersonen har en opfattende (P) eller vurderende (J) livsstil.

Personlighedstesten bliver brugt til at analysere, hvilke svagheder og styrker det enkelte individ har, samt hvordan de bedst kan bidrage i en gruppesammenhæng.^[11]



Figur 1.2: Myers-Briggs personlighedstyper

Når man skal lave heterogene grupper baseret på personlighedstyperne, inddeler man de 16 personlighedstyper i fire forskellige overordnede grupper. De fire grupper er som følgende:

- **Analytikere**

Analytikergruppen består af dem, som er intuitive og træffer beslutninger ud fra deres tanker. Deres fællestræk er N og T. De fire analytiske personlighedstyper er arkitekt (INTJ), logiker (INTP), kommandør (ENTJ) og debattør (ENTP). Analytikerne har en stor lyst til at lære nye ting og er ofte meget fleksible med hvilken metode, der bruges til at lære disse ting. Analytikere har desuden en lyst til hele tiden at forbedre sig selv. Som analytiker kan man komme til at overanalysere alting og komme op med for mange unødvendige idéer, men de sætter sig ikke ned og faktisk udfører deres idéer. Dette betyder, at de funktionelt kan blive udkonkurreret af andre, som ikke tænker lige så meget over idéerne, men blot tager initiativet til at videreføre dem.^[12]

- **Diplomater**

Diplomatgruppen består af dem, som er intuitive og træffer beslutninger ud fra deres følelser. Deres fællestræk er N og F. De fire diplomatiske personlighedstyper er advokat (INFJ), mediator (INFP), protagonist (ENFJ) og aktivist (ENFP). Diplomater er gode til at løse konflikter i gruppen, da de er meget empatiske mennesker. Diplomater har et meget optimistisk livssyn, som smitter af på folk omkring dem. Den negative side ved diplomater er, at de har svært ved at tage rationelle beslutninger. Diplomater bliver hurtigt passionerede omkring emner; selvom deres syn er gyldigt, kan de lettere komme til at fjendtliggøre dem med en anden holdning.^[13]

- **Vagtposter**

Vagtpostgruppen består af dem, som er sansende og har en vurderende livsstil. Deres fællestræk er S og J. De fire vagtpost-personlighedstyper er logistiker (ISTJ), beskytter (ISFJ), leder (ESTJ) og konsul (ESFJ). Vagtposter er meget strukturerede og praktiske i deres arbejde. De er hårdtarbejdende og søger stabilitet. De forventer samme arbejdsmorale og loyalitet fra deres gruppemedlemmer, som de selv giver. En af vagtposternes svagheder er til gengæld at være ufleksible og uacceptable overfor holdninger og synspunkter, som går imod deres egne. De har svært ved at tilpasse sig, hvis ting ikke går efter deres plan.^[14]

- **Udforskere**

Udforskergruppen består af dem, som er sansende og har en opfattende livsstil. Deres fællestræk er S og P. De fire udforskende personlighedstyper er virtuos (ISTP), eventyrer

(ISFP), iværksætter (ESTP) og underholder (ESFP). Udforskere er komfortable med uvished og prioriterer ikke forberedelse højt. Disse personlighedstyper tilpasser sig og overkommer begivenheder, som de forekommer. Udforskere er mestre i forskellige værktøjer og teknikker, som de erfarer ved deres konstante søgen for ny viden. Denne alsidige beslutsomhed betyder dog ikke engagement, da de hurtigt mister interessen for projekter, eftersom de altid søger nye og spændende udfordringer. [15]

Den optimale Myers-Briggs-gruppe ville, ifølge den heterogene gruppesammensætningsteori, være en blanding af de fire typer. De beskrevne grupper har alle svagheder og styrker, som balancerer hinanden ud. Når der dannes heteogene grupper vil man gøre brug af alle gruppemedlemmernes styrker, mens deres svagheder opvejes af hinandens styrker. De forskellige gruppemedlemmer vil også løse konflikter på forskellige måder, og nogle er bedre til at håndtere visse typer konflikter end andre. Dette gør det vigtigt at have forskellige personlighedstyper i en gruppe. Dette betyder ikke, at alle konflikter i en gruppe kan løses, men hver gruppe giver forskellige perspektiver i konflikter, som kan hjælpe med at skabe forståelse overfor hinandens synspunkter og udarbejde en mulig løsning.

1.5 Konflikter i gruppen

Ved gruppearbejde er det næsten en selvfølge, at der vil opstå konflikter i gruppen. Der er flere forskellige faktorer, og dermed også forskellige typer af konflikter. Hovedsageligt er der to pointer omkring disse konflikter. Den første er, at det er hensigtsmæssigt at finde hvilken type konflikt, der er tale om. Den anden pointe er, at konflikterne opstår pga. praktiske, faglige og sociale årsager. [16]

De **praktiske konflikter** opstår, når gruppen har konflikter omkring udefrakommende begivenheder. Dette kan være problemer med mødetider, hvor en gruppe f.eks. har lagt en fast plan, hvor et eller flere af medlemmerne pludseligt har fået ændringer i hverdagen, så individerne ikke længere har mulighed for at overholde den aftalte plan. [16]

Ved **sociale konflikter** omhandler det to eller flere i gruppen, der ikke fungerer sammen socialt. Dette kan skabe splid i gruppen, og er en faktor, som kan forsinke arbejdsprocessen og dermed være kritisk for det videre samarbejde i gruppen. Et eksempel på dette kan f.eks. være, at et medlem har en individualiseret oplevelse af individets arbejdsbyrde. Her ses ofte en følelse af, at medlemmet har skrevet det hele, kommet med alle idéerne, skaffet litteraturen selv og sørget for, at alle mødes. Dette er en naturlig reaktion for mennesker, hvor man netop fokuserer på sin egen arbejdsindsats, og derfor overser, hvad andre gruppemedlemmer har bidraget med. Dette gælder ofte ved en presset situation, hvor gruppen ikke har haft tid til at drøfte disse komplikationer. [16]

Ved **faglige konflikter** forstås det, at gruppen er uenige om det faglige indhold. Dette kan gælde for en gruppe, hvor to eller flere personer mener, at gruppearbejdet skal gå forskellige veje. Denne konflikt er ofte løselig, da konflikten i sin natur lægger op til diskussion, og derfor har gruppen en større mulighed for at løse konflikten. Hvis konflikten ikke løses efter diskussion, er det ofte en lærer eller vejleder, som hjælper gruppen igennem denne konflikt. [16]

En faglig konflikt kan også forekomme, hvis et gruppemedlem vælger at fratage sig arbejdsbyrden, netop fordi der er andre personer til at gøre det, hvilket kaldes for Free-Rider-effekten. Dette ville kunne løses ved, at gruppen sætter nogle faste rammer i form af en gruppekontrakt eller lignende. [17]

Positiv konflikt

Når der opstår en konflikt i gruppen, skal det ikke altid anses som noget udelukkende negativt. Konflikter er en naturlig del af gruppearbejde, da der arbejdes tæt med andre mennesker. Konflikten er først alvorlig, hvis gruppen ignorerer den, eller af anden grund ikke tager den op til diskussion med hensigt på at finde en løsning. Hvis konflikten i stedet bliver løst, kan

konflikten medvirke til en fremdrift for projektet. [16]

Ifølge bogen *Projektarbejdets kompleksitet* [16] menes der, at gruppemedlemmerne faktisk lærer af konfliktløsningsprocessen, og kan derfor bruge det til at komme klogere og stærkere ud af forløbet. [16] [18]

Disse er eksempler på nogle forskellige typer af konflikter og hvorfor, det er relevant, at konflikterne skal løses. Det er derfor vigtigt for en positiv gruppedannelse, at grupperne er i stand til ikke blot at fungere fagligt, men også socialt.

Derfor kan det være en god idé, at inddrage Myers-Briggs eller Belbins teori til gruppedannelse, netop fordi eleverne bliver sat sammen på en sådan måde, hvor de på forhånd har en god opsætning til at løse konflikter - både socialt og fagligt. Disse konflikter kan måske endda mindskes yderligere, hvis eleverne kender til hinandens personlighedstyper. [19] [20].

1.6 Naturfaglige fag som udgangspunkt for gruppedannelse

Vi har besluttet at afgrænse problemet til naturvidenskabelige fag (fysik, kemi, matematik, osv.), fordi gruppearbejde er brugt meget i fysik, kemi og lignende.

Yderligere ses der også et generelt større og oftest længerevarende brug af gruppearbejde inden for naturfaglige fag - specielt på HTX [21], hvor gruppeprojekter er en væsentlig del af undervisningsvejledningen.

Selvom emnet er afgrænset til naturvidenskabelige fag, kan meget af teorien, som bliver beskrevet i denne opgave, også bruges i andre fag og sammenhænge. Dog er det besluttet, at denne opgave vil bearbejde problemet, og eventuelt en løsning dertil, ud fra en naturfaglig synsvinkel.

Vores problem bliver desuden afgrænset til projektarbejde, da grupper til blot et par opgaver ville gøre anvendelsen af programmet til overvejende meget arbejde til noget, som blot vil tage omkring 10 minutter at løse. I projektarbejde er gruppedannelsen meget vigtig, specielt hvis det er et projekt over en længere periode, som ofte ses i naturvidenskabelige fag.

1.7 Spørgeskema

Vi valgte at lave et spørgeskema for at opnå en større forståelse for de studerendes holdninger og erfaringer med gruppedannelse. Da spørgeskemaet var rettet mod gymnasieelever, blev

det sendt til et gymnasium. Dette gymnasium er en kombination af HTX og HHX. Spørgsmålene var lavet i form af multiple-choice samt et enkelt uddybende spørgsmål, da vi ønskede kvantitative svar over kvalitative svar på vores spørgeskema. Målet med spørgeskemaet var desuden at se, om elever også havde oplevet problemer med gruppedannelse, samt hvordan de prioriterer deres gruppe.

I følgende afsnit bliver der forklaret og argumenteret for de forskellige spørgsmål. [22]

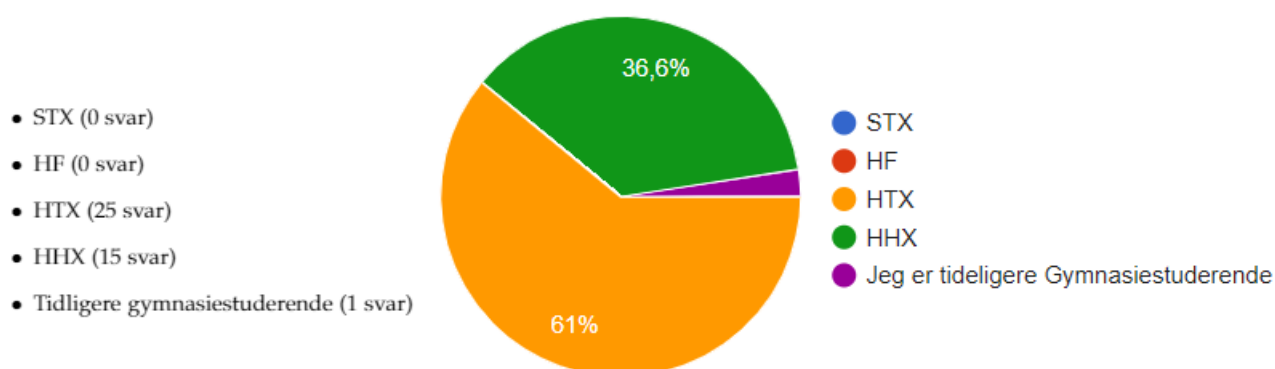
I alt deltog 41 i spørgeskemaet

1.7.1 Spørgsmål 1: Hvilken gymnasial uddannelse går du på?

Det første spørgsmål var, hvilket gymnasium eleven gik på. Der var også mulighed for at vælge, at man var tidligere gymnasiestuderende, i det tilfælde at undersøgelsen endte med at blive sendt til nogen, som ikke længere gik på en gymnasial uddannelse. Grunden til, at spørgsmålet blev stillet i undersøgelsen, var fordi, at vi også ville sikre os, at en HHX-elev ikke endte med at påvirke vores svar på de spørgsmål, som hovedsageligt er rettet mod gymnasiale uddannelser med naturvidenskabelige fag. Grundet dette fokus på resultater fra hovedsageligt naturvidenskabelige elever bliver HHX- og tidligere gymnasieelever automatisk sendt videre til en anden sektion i spørgeskemaet.

Resultaterne er:

41 svar

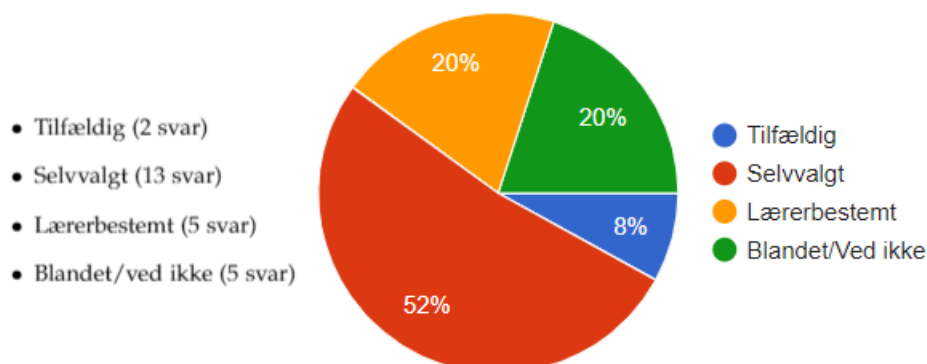


1.7.2 Spørgsmål 2: Hvilken gruppedannelsesmetode gøres der hovedsageligt brug af på dit gymnasium?

I dette spørgsmål bruges tidligere nævnt information om de forskellige gruppedannelsesmetoder: Tilfældig, selvvalgt og lærerbestemt. Der var også en ekstra mulighed for at sige "Blandet/Ved ikke". Resultaterne viser, at der hovedsageligt bliver brugt selvvalgte grupper i gruppedannelsen på gymnasiet (52 %).

Følgende resultater er:

25 svar

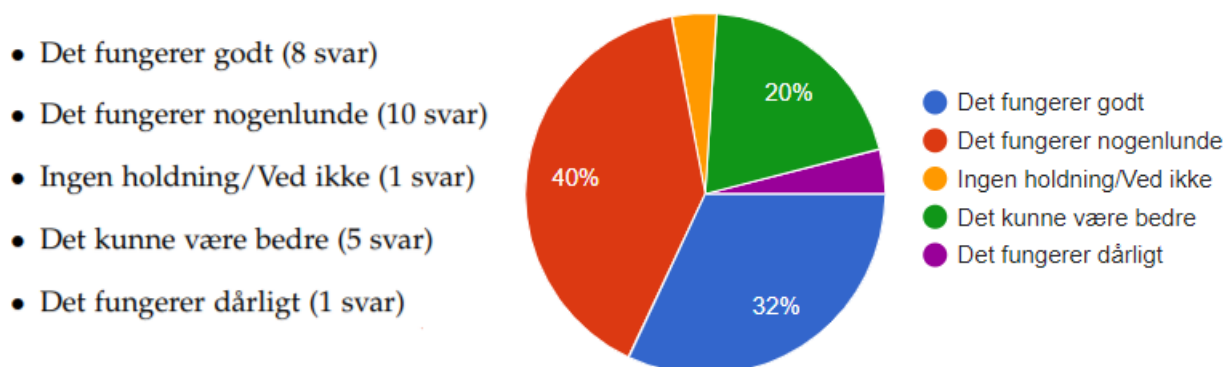


1.7.3 Spørgsmål 3: Hvordan føler du, at gruppedannelse fungerer på dit gymnasium?

Her vurderer den individuelle elev, hvordan deres form for gruppedannelse fungerer. Der kan vælges mellem "Det fungerer godt", "Det fungerer nogenlunde", "Ingen holdning/ved ikke", "Det kunne være bedre", og "Det fungerer dårligt". Da målet er at skabe de bedst mulige grupper, er det også vigtigt at forbedre oplevelsen for alle. Med hensyn til at danne en gruppe er det vigtigt, at alle, der kommer i en gruppe, synes det fungerer godt. At elevernes holdning ligger relativt centreret er et godt tegn, dog ligger enkelte elever i den negative ende, og det er dem, man burde fokusere på.

Resultaterne af spørgsmålet er:

25 svar



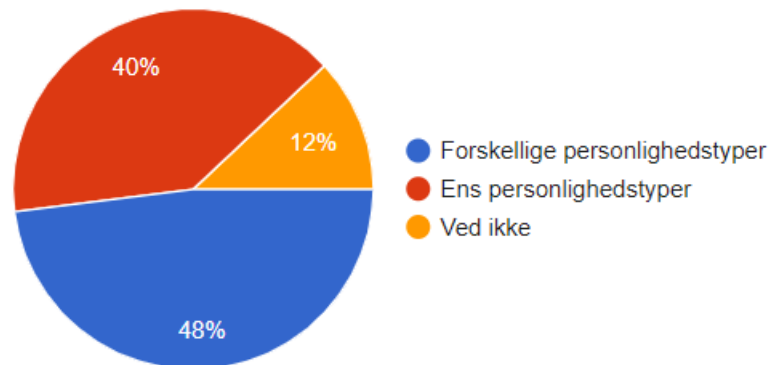
1.7.4 Spørgsmål 4: Hvilken gruppeopbygning føler du, fungerer bedst? Se eventuelt Belbins grupperoller

Formålet med dette spørgsmål er at se, om eleverne allerede har viden inden for gruppedannelse.

Resultaterne er:

25 svar

- Forskellige personlighedstyper (12 svar)
- Ens Personlighedstyper (10 svar)
- Ved ikke (3 svar)



1.7.5 Spørgsmål 5: Hvad prioriterer du højest, det sociale samvær, eller det faglige udbytte i gruppen?

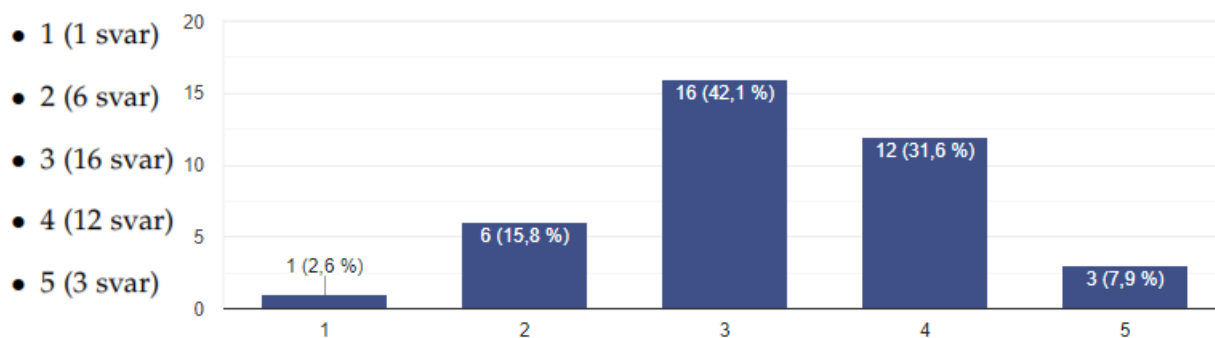
Dette spørgsmål er en skala mellem 1-5, hvor 1 er et fuldt fokus på socialt samvær, og 5 er på fuldt fagligt udbytte. Målet med spørgsmålet er at se, hvad elever på gymnasierne prioriterer højest. Det kan være en fordel at spørge eleverne selv, da spørgsmålet er rettet til at finde deres holdning til, hvad der burde prioriteres i en succesfuld gruppe.

Resultatet giver os en idé om, hvorvidt vi burde lægge vægt på det sociale eller faglige aspekt af gruppedannelsen. Det er ikke nødvendigvis vores direkte mål, at lave det præcist efter resultatet på spørgsmålet, men det kan fungere som et fint udgangspunkt eller guideline.

Resultaterne er:

38 svar

Fra 1-5 hvor socialt samvær er 1 og fagligt udbytte er 5

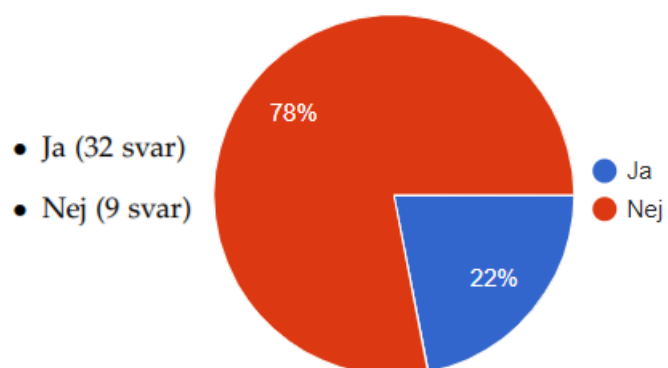


1.7.6 Spørgsmål 6: Har du kendskab til nogen grupperolletests?

Dette er et simpelt ja/nej spørgsmål, som blot sorterer dem fra, der ikke kender til nogle grupperolletests. Spørgsmålet sender altså alle, der svarer ja, videre til spørgsmål 7.

Resultat blev:

41 svar



1.7.7 Spørgsmål 7: Hvilke grupperolletests kender du til?

I dette spørgsmål undersøges, hvilke grupperolletests de elever, der svarede ja på foregående spørgsmål kender til. Der svares på dette spørgsmål ved at indtaste svaret i en tekstboks.

Der er 2 pointer med spørgsmålet. Det første er at få inspiration til eventuelle andre teorier, der muligvis kan gøres brug af. Den anden pointe er at se, om eleverne allerede har viden om Belbin eller Myers-Briggs, som vi analyserer i rapporten.

Dette vil også kunne give os en idé, om der bliver brugt nogen grupperollebestemte grupper allerede nu på de adspurgte gymnasier.

Resultatet er:

7 svar

- "Kan ikke huske hvad de hedder."
- "Belbin."
- "Kan ikke huske dem ved navn men der er en med 4 bokstaver som indikere hvilken specifik rolle du hører til."
- "<https://www.finduddannelse.dk/educationtest/start/129>"
- "Kan ikke huske."
- "Kan ikke huske."
- "Starter, afslutter, formulerede."

1.7.8 Refleksion

På trods af nogle af resultaternes brugbarhed er der stadigvæk problemer med selve spørgeskemaet. Et af problemerne er, at der under spørgsmålet angående elevernes holdninger til, hvordan gruppedannelsesprocessen var gået, er flere svarmuligheder, men ingen uddybende spørgsmål bliver præsenteret. Det er derfor ikke tydeligt, om det er det faglige eller sociale udbytte, der er et problem, eller om det er et helt andet problem til at starte med.

Antallet af svarende elever burde have været større for at opnå et mere repræsentativt resultat. Vi kunne have sendt spørgeskemaet ud til flere gymnasier.

Under det første spørgsmål om uddannelse sorteres tidligere gymnasieelever fra; de fik altså kun spørgsmål 6 og 7. Deres besvarelser kunne have været nyttige, da nyligt dimitterede elever har en vis erfaring - samt har haft tid til at reflektere yderligere - noget som dem, der stadigvæk er midt i uddannelsen, muligvis mangler. Deres viden giver derfor et bedre indblik i, hvordan gruppedannelsen i gymnasiet oprigtigt fungerer, frem for dem, der lige er startet på deres første år. De gymnasieelever, som er fra HHX, er sorteret fra, fordi denne gymnasiale uddannelse ikke indeholder naturfaglige fag.

Det fjerde spørgsmål fra spørgeskemaet kunne have været formuleret bedre. Der foreslås, at eleven selv ser på Belbins grupperoller, men svarmulighederne vedrører, hvorvidt eleven mener, at det er bedst med ens eller forskellige personlighedstyper. Belbins grupperoller er ikke en personlighedstypetest, men derimod en test, som definerer en persons grupperolle i gruppearbejde.

I det 7. spørgsmål ville det også have været en fordel at tilføje 3 forudbestemte svarmuligheder, som kunne lyde på "Kan ikke huske navnet", "Belbin", "Myers-Briggs" og en sidste mulighed med en tekstboks til at skrive et eventuelt navn på en anden grupperolletest. Dette kunne dog have ledt eleverne til at vælge disse muligheder. I tilfældet, hvor vi kun har syv svar, er det ikke det store problem, men havde vi haft mange flere svar, ville det være smart at have et tal på "Kan ikke huske" i stedet for 10 forskellige måder, det er skrevet på. Det samme gælder for Belbin og Myers-Briggs, da det eneste ukategoriserede svar, der skal læses, ville være nye forslag. Vi fik en ny grupperolletest ud af spørgsmålet, dog var den designeret til teamroller i et arbejdsmiljø. Det er også meget begrænset hvad angår kilder og teori bag selve testen - derfor er der ikke kigget yderligere på denne test.

Spørgsmålet har mange problemer, og vi har derfor valgt ikke at arbejde videre med spørgeskemaet i denne rapport.

2. Problemformulering

Gruppearbejdet forløber nemmere, hvis individet er bevidst om sin egen og andres roller i gruppen. Dr. Meredith Belbin udtaler: "I en gruppe er det balancen, der er afgørende. Det, der er brug for, er ikke velafbalancerede individer, men individer, som er godt afbalancerede i forhold til hinanden. På den måde kan menneskers svage sider stives af, og de stærke udnyttes fuldt ud".^[23]

Hypotesen omkring udbyttet fra de eksisterende løsninger viser sig at være sand. Der vil næsten altid være elever, der får mindre ud af den gruppe, de er blevet tildelt, end andre elever. Det vil sige, at individet, af ikke-konsekvente årsager, ikke matcher med den tildelte gruppe. Har man i tankerne, hvordan dette individ arbejder i forhold til deres grupperolle, kan man muligvis matche dem med andre, der afhjælper deres mangler. For at danne en stærkere og mindre konfliktpræget gruppe, skal medlemmer, ifølge Belbin, parres således, at de afdækker så mange af de 9 roller som muligt.

Hvordan kan software forbedre gruppedannelsesprocessen i gymnasiet, således der opnås bedre faglige resultater og afhjælpes konflikter, med udgangspunkt i elevers grupperoller og ambitionsniveau?

3. Problemløsning

På baggrund af problemanalysen vil vi nu forsøge at løse problemet givet i problemformuleringen. Dette gøres ved først at opstille løsningsforslag og herefter en kravspecifikation til et program.

3.1 Løsningsforslag

Løsningen, som vi er nået frem til, er i form af et program i programmeringssproget C, som skal være i stand til at sammensætte arbejdsgrupper til en gymnasieklasse. Læreren indtaster alle eleverne i en tekstfil samt indstiller forskellige parametre, der skal danne basis for, hvilken type gruppedannelse, der ønskes. Der skal både indgå faglige og sociale parametre; eleverne tager en Belbins grupperolletest, hvor de får tildelt 3 grupperoller. Elevernes roller skal herefter indtastes i tekstfilen, hvorved programmet sorterer efter hensigten, at alle grupper får tildelt så mange forskellige grupperoller som muligt. De sociale grupper skal primært sammensættes ud fra elevernes egne ønsker, specifikt hvem de gerne vil og ikke vil være i gruppe med. Den tredje og sidste parameter er ambitionsniveau, som skal have indflydelse på gruppedannelsen i både de faglige og sociale grupper.

Parametrene er:

- Grupperoller (Elever tager Belbin-test)
- Ambitionsniveau (på en skala fra 1-5)
- Elevernes egne ønsker
 - Ønske om 3 elever man GERNE vil arbejde sammen med
 - Ønske om 1 elev man IKKE vil arbejde sammen med

Vi har valgt at bruge disse parametre fordi:

Grupperoller: Vi har igennem problemanalysen haft fokus på Belbins grupperoller og Myers-Briggs personlighedstyper. Vi tager udgangspunkt i Belbins grupperoller, eftersom denne teori bliver brugt til at danne projekt-teams på arbejdspladser, og der har tidligere været samarbejde mellem Belbins teorier og undervisninginstitutioner.^[24] Vi har dog valgt ikke at gå videre med Myers-Briggs personlighedstyper, da deres hovedformål ikke er at danne grupper, men at skabe større forståelse mellem individer - ofte i en allerede dannet gruppe.

Ambitionsniveau: I et interview med gymnasielærer Nanna Louise Ben-Or, se bilag [A](#), fortæller hun, at hendes go-to-metode til gruppedannelse er, at eleverne skriver deres ambitionsniveau på tavlen, hvorefter de sorteres således de bliver grupperet med elever med ens ambitionsniveau. Hun har erfaret, at nogle elever flytter sig fagligt, og at det varierer fra hver gruppedannelse og hvilket fag, det omfatter. Dog kan sociale relationer mellem eleverne spænde ben for den type gruppedannelse. Hun opfordrer til, at der skal være lærerstyring, hvis sådanne ambitionsniveau-fokuserede grupper skal dannes.

Vi overvejede originalt både ambitionsniveau og karaktergennemsnit som mulige sorteringsfaktorer, men ved sortering efter karakter opstår problemet om, hvilken sortering der ville være mest fordelagtig. Grupperne kunne sammensættes med et bestemt karaktergennemsnit som mål i en given gruppe, men det ville betyde, at 12- og 10-talselever konsekvent ville blive sat i gruppe med 02- og 4-talselever, hvilket kunne have en negativ effekt på de førnævnte, hvis de følte sig tvunget.

Elevernes ambitionsniveau virker kun som en troværdig parameter, hvis eleverne selv er ærlige, når de skriver det på tavlen. Det kan skabe problemer, hvis en elev f.eks. skriver et for højt tal, hvorefter det ikke stemmer overens med den arbejdsindsats, der i gruppen forventes af dem. At eleverne offentligt skal skrive deres ambitionsniveauer kan bl.a. være skyld i disse upræcisheder, da få sikkert vil indrømme, at deres ambitionsniveau ligger lavt overfor resten af klassen. Dette burde derfor eventuelt sendes til læreren direkte, så sådanne situationer undgås.

Elevernes egne ønsker: Som nævnt af sociolog Simon Søndergaard Christensen, se bilag [A](#), er det væsentligt, at eleverne føler, at de har haft tilstrækkelig indflydelse på gruppedannelsen. En mangel på indflydelse kan have en negativ effekt på elevernes engagement og samarbejdsvillighed.

3.2 Kravspecifikation

I kravspecifikationen er der udtænkt et overblik over de kriterier, der skal til for at vi kan opbygge et program, som kan danne grupper. Herunder indgår der nogle krav om input fra læreren.

- Input fra brugeren (læreren) skal ske ved hjælp af en tekstfil, som indeholder følgende:
 - Antal ønskede grupper
 - Ønsket gruppedannelsesparameter (social eller faglig)
 - Herefter én elev per linje, der indeholder de nødvendige informationer om eleven, herunder:

- * Elevens navn
- * Elevens ambitionsniveau
- * Elevens tre mest fremtrædende grupperoller
- * Tre elever, som eleven ønsker at være i gruppe med
- * Én elev, som eleven *ikke* ønsker at være i gruppe med

Antal ønskede grupper er væsentlig, fordi det bruges i løsningen til at beregne hvor mange elever, der skal være i hver gruppe. Mulighed for social eller faglig gruppedannelse er også op til læreren, og dertil bliver information om, hvad hver gruppedannelsesmetode indebærer lagt tilgængeligt i input-filen. Denne fil er vedlagt programmet og indeholder også en skabelon samt vejledning til, hvordan læreren skal skrive eleverne ind; se bilag [F](#). Programmet læser linjerne, som repræsenterer hver elev, og indlæser denne data, som efterfølgende bearbejdes til gruppedannelse.

- Der skal laves et struct "student", hvor informationerne om eleverne læses over i. Denne struct skal indeholde:

```

- Name[30] (char)
- ambitionLevel (int)
- wishedAmount (int)
- wishedBy[10][30] (char)
- roles[3] (enum belbin)
- doWant[3][30] (char)
- notWant[30] (char)
- isInGroup (bool)

```

Disse struct-members vil blive gennemgået mere fyldestgørende i sektion 3.3

- Programmet skal skrive resultatet til en ny tekstfil med formatet:

GRUPPE 1:

Navn 1 (ambitionsniveau, grupperoller, ønsker)

Navn 2 (ambitionsniveau, grupperoller, ønsker)

...

GRUPPE 2:

Navn 1 (ambitionsniveau, grupperoller, ønsker)

Navn 2 (ambitionsniveau, grupperoller, ønsker)

...

GRUPPE 3:

... Vi har valgt at udskrive de dannede grupper i en separat output-tekstfil.

Hvis informationen om eleverne ikke bliver ændret fra første gruppedannelse, kan det ikke bruges til at danne nye grupper. Det er væsentligt, at eleverne ændrer noget information, evt. nyt ambitionsniveau eller andre elevønsker/-fravalg, hvis der ønskes andre grupper.

- Hvis antallet af elever ikke går op med det ønskede antal elever pr. gruppe, skal programmet selv dele "resteleverne" ud på de ellers fyldte grupper.
- **Fejlhåndtering:**
 - Inputfil ikke fundet
 - * Fejl meldes til brugeren, hvorefter en ny inputfil automatisk genereres.
 - Elevformat i inputfil passer ikke med det forventede format
 - * I dette tilfælde stopper alle operationer, og der udskrives en fejl til brugeren, der viser hvilken linje samt sted, hvor fejlen er opstået, så brugeren let kan rette den.

3.3 Beskrivelse af software

Til produktet anvendes standarden ANSI C89, som har været anvendt igennem det sidefølgende kursus *Imperativ Programmering* (IMPR).

Programmet er opbygget af 6 kildefiler; `main.c`, `readFile.c`, `groups.c`, `belbinGroups.c`, `sortWishes.c` og `strlwr.c`

main.c er en overfladisk fil, der sammenkobler alle de andre funktionsfiler. Den fungerer som skelettet for programmet.

readFile.c indeholder de funktioner, der håndterer indlæsning af inputfilen samt generering af en ny inputskabelon, hvis denne ikke findes i forvejen.

groups.c indeholder de funktioner, der håndterer grupperne selv. Dette inkluderer oprettelse af grupper, tilføjelse af elever til grupper, tælling af antal elever, der allerede er i en gruppe, samt at udskrive de endelige grupper til outputfilen. Det er desuden her, struct `student` og enum `role` er definerede.

strlwr.c er en hjælpefunktion, eftersom den normale `strlwr` ikke er en del af standard library (`stdlib`) på visse computere.

belbinRoles.c indeholder de funktioner, der har at gøre med at danne grupper med fagligt fokus, sådan at der er flest muligt forskellige grupperoller i hver gruppe.

sortWishes.c har de funktioner, der håndterer den sociale inddeling af eleverne i grupper, hvor der er fokus på de indtastede ønsker, eleverne har.

3.3.1 Struct student

I alle funktioner har vi så vidt som muligt markeret input-parametre med `const`.

Vi har initialiseret et struct kaldet `student`, som indeholder samtlige input-variable, der skal ligge lager til input-data fra den tekstfil, læreren skal administrere, som man kan se i kodeudsnit [3.1](#)

Kodeudsnit 3.1: struct student

```
29 struct student
30 {
31     char name[30];
32     int ambitionLevel; /*from 1 to 5*/
33     int wishedAmount;
34     char wishedBy[10][30];
35     role roles[MAX_ROLES];
36     char doWant[3][30];
37     char notWant[30];
38     bool isInGroup;
39 };
```

- Første struct-member, name, er af typen char array, med 30 karakterer allokeret, hvilket burde kunne indeholde et fornavn og eventuelt efternavn.
- Elevens ambitionsniveau har vi kaldt ambitionLevel, og den er af typen int, da vi regner med, at læreren indtaster et heltal fra 1 til 5 svarende til ambitionsniveauet.
- Memberen roles er af typen enum role, som består af alle 9 Belbins grupperoller. MAX_ROLES er lig 3, da hver elev skal have deres 3 mest fremtrædende grupperoller med i tekstfilen.
- Vi har to char arrays: Ét til elevernes ønske om samarbejdspartnere, som kan tage 3 elever, og et til fravalg af én elev. Igen er der allokeret 30 karakterer til fornavn og eventuelt efternavn.
- Til sidst i struct student har vi en member isInGroup af typen bool, som tjekker, om en elev allerede er i en gruppe, så eleverne ikke uddeles på mere end én gruppe hver.

3.3.2 Tekstfil-input

De første 38 linjer af tekstfilen er en input guide, der viser, hvordan data skal udfyldes i bunden af dokumentet samt accepterer input for gruppeantal og gruppedannelsesmetode. På linje 25 indtastes antallet af ønskede grupper. Dette løses ved hjælp af en funktion, der tjekker, hvilket tal der er skrevet i linje 25, og returnerer dette. Hvis der indtastes under 3 giver det en fejlbesked, der prompter at der ikke kan være mindre end 3 grupper, samt hvis der er skrevet ingenting, promptes der, husk at skrive ønskede gruppetal.

På linje 28 og 29 skal læreren specificere hvilken sorterings metode, altså hvilket gruppefokus (fagligt eller socialt) skal bruges. Der skal indtastes et x i netop én af boksene. Hvis der indtastes intet, eller andet end et x, giver det en fejl besked i prompten; sæt et enkelt kryds.

På linje 39, dvs. efter læreren har fået instrukster til, hvordan de nødvendige informationer til hver elev skal skrives ind, skal eleverne skrives ind på følgende måde:

- *Format*: Navn, Ambitionsniveau, Rolle 1, Rolle 2, Rolle 3, Ønske 1, Ønske 2, Ønske 3, Fravalg.
- *Eksempel*: Peter, 2, iga, ide, kon, Henrik, Sophie, Arne, Anne.

Samtlige elev tekststrengene bliver læst ind i structen student ved brug af en fscanf funktion. se kodeudsnit [3.2](#)

Kodeudsnit 3.2: readFile

```

96 | for(i = 0; i < numberOfStudents; i++)
97 | {
98 |     scanRes = fscanf(inFP, " %[^,], %d, %[^,], %[^,], %[^,], %[^,],
    |         %[^,], %[^,], %[^.].",
99 |         studentList[i].name, &studentList[i].ambitionLevel,
    |         rolesStr[0], rolesStr[1],
100 |         rolesStr[2], studentList[i].doWant[0], studentList[
    |         i].doWant[1],
101 |         studentList[i].doWant[2], studentList[i].notWant);
102 |     studentList[i].isInGroup = false;

```

Funktionen læser til kommaet i tekststrengene, se linje 98 kodeudsnit [3.2](#). Samtlige indlæsninger, der ikke skal formateres som heltal, bliver læst af en *format specifier*, som lagrer teksten i en streng. Samtlige struct members af typen char array, får tildelt en streng hver. Undtagen ved det andet tekst element, se %d i fscanf. Denne integer lægges over i &studentList[i].ambitionLevel, og holder på hvilket som helst heltal. Men hvis heltallet ikke er mellem 1 til 5, promptes der en brugervenlig fejlmeddelelse i output filen.

Kodeudsnit 3.3: ambitionsNiveau

```

96 | if(studentList[i].ambitionLevel > 5 || studentList[i].ambitionLevel < 1)
97 | {
98 |     printf(" *Det indtastede Ambitionsniveau er ikke imellem 1 og 5!
    |         se linje %d!\n", i + LINES_SKIPPED+1);
99 |     printf("%s\n", studentList[i].name);
100 |     ambRes++;
101 | }

```

I fscanf-funktionen itererer vi som sagt samtlige elevstrengene, og hvis der findes elever, der har fået tildelt et tal større end 5 eller mindre end 1 (se kodeudsnit [3.3](#) linje 96), udskrives der en besked, der netop fortæller, at ambitionsniveau er tastet ind, så programmet ikke kan læse

det. ambRes, se linje 100.

Ligeledes udskrives der fejlmeddelelser til fejl ved indlæsning af gruppe fokus se kodeudsnit 3.4

Kodeudsnit 3.4: gruppeFokus

```
77 | printf("[%c] [%c]\n", optionFirst, optionSecond);
78 |     printf(" * Fejl i linje 28/29 - prioritetsmode. Saet et enkelt kryds (x)
    |         !\n");
79 |     return error;
```

3.3.3 Dannelse af faglige grupper

Vælger læreren at der skal dannes fagligt orienterede grupper, kaldes funktionen sortBelbin. Strukturen af sortBelbin er vist med et flowdiagram på fig. 3.1

Funktionen sortBelbin er af typen void, da dens output er et to-dimensionalt array af typen struct student, og det derfor foregår igennem et output-parameter. sortBelbin har i alt fem parametre, hvilket man også kan se i kodeudsnit 3.5

Kodeudsnit 3.5: sortBelbin prototype

```
1 | void sortBelbin(student studentList[], int rolesCount[9][2], const int
  |     numberOfStudents, const int groupAmount, student **groups)
```

Det første parameter er student studentList[], der er et array af alle elever der er indlæst fra inputfilen og som i øjeblikket bliver håndteret af programmet. Denne er ikke decideret et output-parameter, men arrayet bliver sorteret og der bliver ændret på elevernes gruppestatus løbende, hvilket gør at den ikke kan være const.

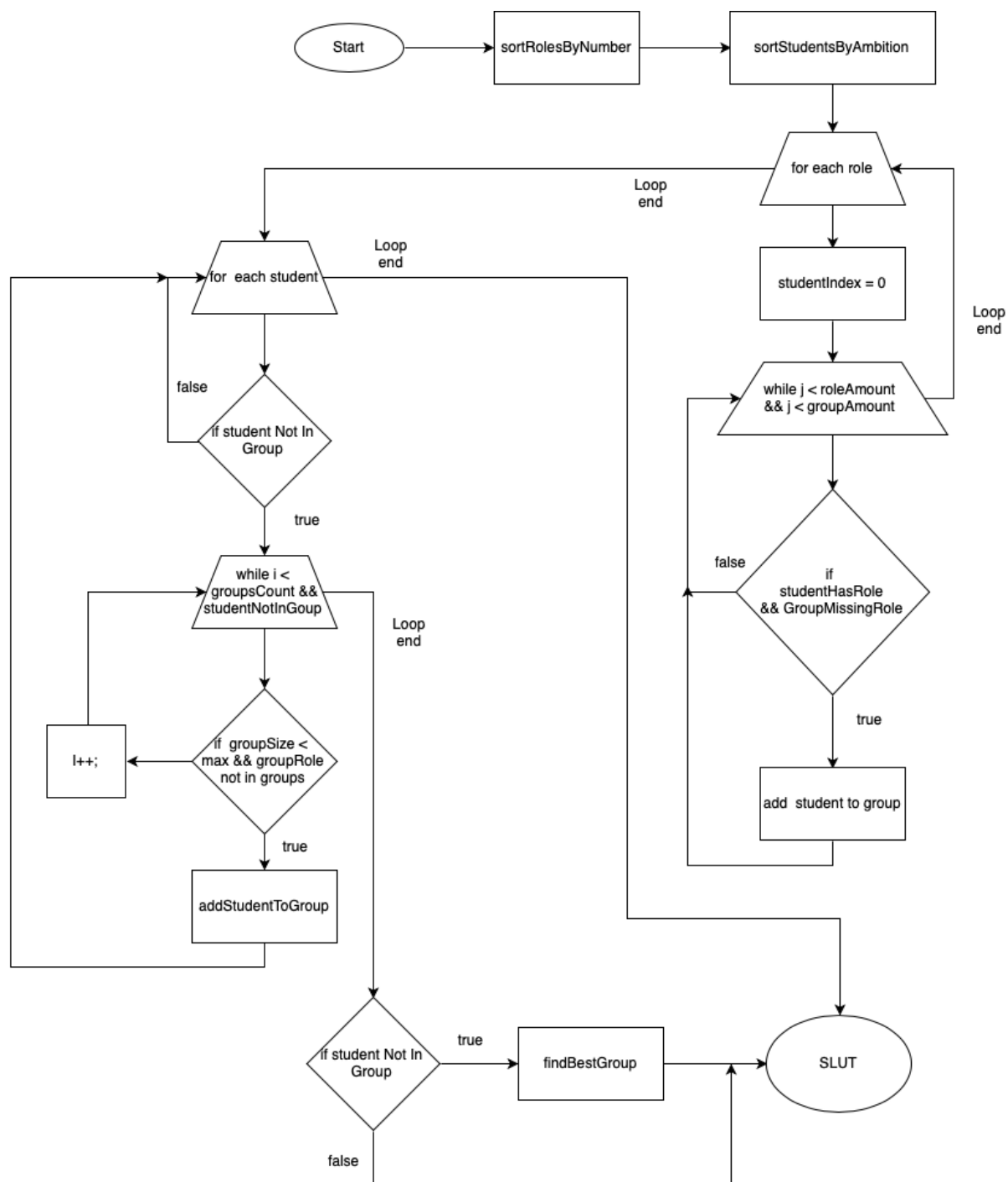
Andet parameter er int rolesCount[9][2], der er et to-dimensionalt array der holder grupperollefordelingen blandt eleverne. Hver af de ni kolonner har 2 rækker, én der holder hvilken grupperolle det er ved hjælp af enum role, samt hvor mange elever der har denne grupperolle. Dette er ligeledes ikke et output-parameter, men den bliver også sorteret og kan derfor heller ikke være const.

Tredje parameter const int numberOfStudents er et input-parameter der holder antallet af elever der arbejdes med.

Det sidste input-parameter const int groupAmount holder antallet af grupper der skal laves. Desuden har funktionen variablen int studentPerGroup der er defineret som studentPerGroup = numberOfStudents/groupAmount og altså antallet af elever i hver gruppe.

Funktionen sortBelbin har 3 stadier:

Figur 3.1: Flowdiagram faglige grupper



1. Sorter roller og elever
2. Fordel én af hver rolle i hver gruppe
3. Fordel resten af eleverne

Det første stadie er blot forberedende, der består af to kald af `qsort`, hvor `rolesCount` bliver sorteret således at de grupperoller der er færrest elever der har, kommer først. Desuden bliver `studentList` sorteret efter ambitionsniveau sådan at dem med højest ambitionsniveau står først.

Andet stadie har i sig selv to dele. I første del tages der udgangspunkt i hvilke grupperoller der er færrest af blandt eleverne, og fordeler dem således at de ikke kommer i samme gruppe, se kodeudsnit [3.6](#).

Kodeudsnit 3.6: Roller fordeles del 1

```

8 | for (i = 0; i < 9; i++)
9 | {
10 |     int studentIndex = 0;
11 |     j = 0;
12 |     while(j < rolesCount[i][1] && j < groupAmount && studentIndex <
        numberOfStudents)
13 |     {
14 |         if(!studentList[studentIndex].isInGroup && studentHasRole(rolesCount[
            i][0], &studentList[studentIndex]) && groupMissingRole(groups[j],
            rolesCount[i][0], studentPerGroup))
15 |         {
16 |             addToGroup(groups[j], &studentList[studentIndex], studentPerGroup);
17 |             j++;
18 |         }
19 |         studentIndex++;
20 |     }
21 | }

```

I anden del af dette stadie loopes der igennem alle eleverne og hvis eleven ikke allerede er i en gruppe, bliver eleven sat i den første gruppe, der mangler én af denne elevs grupperoller, se kodeudsnit [3.7](#). På den måde bliver der så vidt som muligt fordelt mindst én af alle grupperoller i hver gruppe. Desuden kommer grupperne sekundært til at være arrangeret sådan at hver gruppe så vidt som muligt vil have det samme ambitionsniveau. Dette sker fordi at arrayet `studentList` er sorteret efter ambitionsniveau, så når der i hvert trin loopes igennem dette array, f.eks i kodeudsnit [3.6](#), kommer den første elev i arrayet med den grupperolle, og derfor også den elev med højest ambitionsniveau der har den pågældende grupperolle, i gruppe 1. Den næste elev i arrayet der har denne grupperolle, og derfor den elev med det næsthøjeste ambitionsniveau, kommer i gruppe 2, osv. Når alle der har denne grupperolle er

fordelt, startes der fra index 0 i `studentList` igen, så den første elev i arrayet der har rolle nummer to, og derfor også den elev med det højeste ambitionsniveau og rolle to, kommer i gruppe 1. Den næste i arrayet kommer i gruppe 2 osv.

Gruppe 1 kommer derved, så vidt som muligt, til at bestå af de elever der inden for hver rolle har det højeste ambitionsniveau.

Kodeudsnit 3.7: Roller fordeles del 2

```

22 for(i = 0; i < numberOfStudents; i++)
23 {
24     if(!studentList[i].isInGroup)
25     {
26         j = 0;
27         while(j < groupAmount && !studentList[i].isInGroup)
28         {
29             if(studentsInGroup(groups[j], studentPerGroup) < studentPerGroup)
30             {
31                 k = 0;
32                 while (k < MAX_ROLES && !studentList[i].isInGroup)
33                 {
34                     if(groupMissingRole(groups[j], studentList[i].roles[k],
35                                         studentPerGroup))
36                     {
37                         addToGroup(groups[j], &studentList[i], numberOfStudents
38                                     );
39                     }
40                     else
41                     {
42                         k++;
43                     }
44                 }
45             }
46             j++;
47         }
48         if (!studentList[i].isInGroup) {
49             j = findBestGroup(&studentList[i], groups, groupAmount,
50                             studentPerGroup + 1);
51             addToGroup(groups[j], &studentList[i], studentPerGroup + 1);
52         }
53     }
54 }

```

I det sidste stadie af `sortBelbin`, tilføjes eventuelle resterende elever til de grupper hvor de passer bedst ind. Dette bliver afgjort ved hjælp af funktionen `findBestGroup` der returnere

det index til groups den bedste gruppe har, se linje 47 i kodeudsnit 3.7. Den bedste gruppe bliver afgjort ud fra følgende prioritet:

1. Antal elever i gruppe
2. Antal grupperoller eleven har, som gruppen mangler
3. Ambitionsniveau

Denne prioritet er valgt fordi det på dette stadie er vigtigst at der i alle grupper er ca. lige mange elever. Er det tilfældet at der er lige mange elever i hver gruppe, skal en elev tilføjes i den gruppe der mangler flest af de grupperoller den elev har. Er alle grupperoller dækket i alle grupper, bliver eleven tilføjet i den gruppe, hvor det gennemsnitlige ambitionsniveau er tættest på elevens eget ambitionsniveau. Se denne fremgangsmåde i kodeudsnit 3.8.

Kodeudsnit 3.8: findBestGroup

```
1 int findBestGroup(const student *inStudent, student **groups, const int
  groupAmount, const int groupSize)
2 {
3     int i,j,res = 0, bufferA,bufferB;
4     double dBufferA, dBufferB;
5     for(i = 1; i < groupAmount; i++)
6     {
7         bufferA = studentsInGroup(groups[i],groupSize) - studentsInGroup(
            groups[res],groupSize);
8         if(bufferA < 0)
9         {
10             res = i;
11         }
12         else if(bufferA == 0)
13         {
14             bufferB = 0;
15             for(j = 0; j < MAX_ROLES; j++)
16             {
17                 bufferA += groupMissingRole(groups[res],inStudent->roles[j],
                    groupSize);
18                 bufferB += groupMissingRole(groups[i],inStudent->roles[j],
                    groupSize);
19             }
20             bufferA -= bufferB;
21             if(bufferA < 0)
22             {
23                 res = i;
24             }
25     }
```

```

26         else if(bufferA == 0)
27         {
28             dBufferA = fabs(inStudent->ambitionLevel -
                             averageAmbitionInGroup(groups[res], studentsInGroup(groups
29             [res], groupAmount)));
30             dBufferB = fabs(inStudent->ambitionLevel -
                             averageAmbitionInGroup(groups[i], studentsInGroup(groups[i]
31             [, groupAmount)));
32             dBufferA -= dBufferB;
33             if(dBufferA > 0)
34             {
35                 res = i;
36             }
37         }
38     }
39     return res;
}

```

3.3.4 Dannelse af sociale grupper

sortWishes er funktionen som tager sig af opdelingen af grupper baseret primært på elevernes ønsker og fravalg. Hver elev har skrevet 3 ønsker, som er elever der er blevet ønsket af den bestemte elev. Eleverne har også et fravalg, af én elev/person som ikke ønskes i personens gruppe. Grupperne bliver dermed dannet baseret på disse parametre. I nogle tilfælde skal der dannes grupper som har en rest af elever. Det vil sige at antal gruppe og antal elever går ikke op i hinanden. For eksempel en gruppe på 11 elever skal deles over to grupper. Der er en rest på 1 elev. Disse elever bliver prioriteret ved hjælp af det gennemsnitlige ambitionsniveau. Funktionen er også sat op således, at dem som ikke er blevet ønsket af nogen, ikke er blevet sat i en gruppe på dette tidspunkt af programmet (se det fulde flowdiagram af sortWishes her) [\[25\]](#). Funktionen tager kun brug af en anden funktion, som specifikt er lavet til sortWishes. Funktionens navn er wishedCmp, og bliver brugt som comparefunktion til qsort. Funktionen tager 4 parametre:

1. student studentList[]
2. int numOfStudents
3. int maxGroups
4. student **group (pointer)

Kodeudsnit 3.9: sortWishes prototype

```
1 || void sortWishes(student studentList[], int numOfStudents, int maxGroups,
   || student **group)
```

Hvor studentList, et array af student structs, som er alle vores elever der er blevet skrevet i input-tekstfilen. numOfStudents er antallet af studerende af elever i alt i arrayen, for at holde styr på diverse loops, og hvornår programmet skal hoppe ud af specifikke løkker. maxGroups er antallet af grupper som maks skal dannes, og er også det input som er angivet i input.txt. **group er pointere til vores grupper, group er en pointer til en 2-dimensional array, hvor første dimension er grupper, og anden dimension er de studerende i de forskellige grupper. Dermed kan vi loope igennem forskellige grupper, og strukturen er også nemmere at holde styr på. Størrelsen på det todimensionale array er baseret på hvad maxGroups er, og er udregnet tidligere i programmet.

Funktionen starter med at udregne hvor mange gange de forskellige studerende er blevet ønsket af andre elever. For opsummering, ser student structet ud således:

Kodeudsnit 3.10: struct students

```
1 || struct student
2 || {
3 ||     char name[50];
4 ||     int ambitionLevel; /*from 1 to 5*/
5 ||     int wishedAmount;
6 ||     char wishedBy[10][30];
7 ||     role roles[MAX_ROLES];
8 ||     char doWant[3][30];
9 ||     char notWant[30];
10 ||    bool isInGroup;
11 ||    int wishedAmount;
12 ||    char wishedBy[20][30];
13 || };
14 || typedef struct student student;
```

En hurtig start til at håndtere og tjekke nemmere hvor mange personer der er i gruppen, laver vi et array af integers, som holder antal elever i gruppen. Tallenes position i arrayet svarer til gruppens nummer i gruppearrayen.

Kodeudsnit 3.11: array groupSizes

```
1 || int groupSizes[maxGroups];
```

Derefter en for-løkke som sætter alle værdierne i groupSizes til 0 og også wishedAmount for hver student = 0.

Kodeudsnit 3.12: Set Group Sizes loop

```
27 || for(i = 0; i < numOfStudents; i++)
28 || {
29 ||     studentList[i].wishedAmount = 0;
30 ||     studentList[i].isInGroup = false;
31 || }
32 || for(i = 0; i < maxGroups; i++)
33 || {
34 ||     groupSizes[i] = 0;
35 || }
```

Vi laver et for loop med variablen i, som itererer igennem alle elever, altså i < numOfStudents. Derefter kører vi et nested for loop igennem elever igen ved navn j. Dette for loop har samme conditions som det første, bare med variablen j. Siden alle elever har 3 ønsker, laves der dermed også endnu et nested for loop. Dette kører imens k er mindre end 3. Siden alle elever har 3 ønsker hver, og array index starter på 0. Dermed er de 3 ønsker fordelt over 0, 1 og 2. Der bruges strcmp (string compare) til at sammenligne hvert ønske med student på plads i's navn. Hvis de passer, bliver student[i].wishedAmount lagt til en (student[i].wishedAmount++). Denne sammenligning sker med alle studerendes ønsker, og bagefter går løkken videre til den næste studerende, og tæller deres gange de er blevet ønsket op.

Kodeudsnit 3.13: Get wishesAmount

```
37 || /* Count the times the different students have been wished */
38 || for(i = 0; i < numOfStudents; i++)
39 || {
40 ||     for(j = 0; j < numOfStudents; j++)
41 ||     {
42 ||         for(k = 0; k < 3; k++)
43 ||         {
44 ||             if(strcmp(studentList[i].name, studentList[j].doWant[k]) ==
45 ||                 0)
```



```

46 |         strcpy(studentList[i].wishedBy[studentList[i].
47 |             wishedAmount], studentList[j].name);
48 |         studentList[i].wishedAmount++;
49 |     }
50 | }
51 | }

```

Dette bruges til at sortere elever i `studentList` efter deres mængde af gange de er blevet ønsket. Det gøres af to grunde: Den første grund er at vi dermed ikke får forskellige resultater, baseret på hvilken position de er blevet placeret i listen. F.eks. hvis eleverne selv har fået besked på at skrive sig op i inputfilen, så er der ikke nogen konkret fordel i at være den første, eller den sidste i filen. Da uanset hvilken placering de er i input filen, kommer resultatet altid til at være baseret på hvor mange gange de er ønsket af andre elever, og ikke baseret på rækkefølgen i filen.

Den anden grund er at vi starter med de personer som har færrest mængde gange de er blevet ønsket. Vi bruger `qsort` til at sortere dem baseret på deres `wishedAmount` variabel i deres struct `student`. Derefter starter vi fra den laveste til den højeste. Vores compare funktion er `wishedCmp`.

Kodeudsnit 3.14: `qsort` wishesAmount

```

53 | qsort(studentList, numOfStudents, sizeof(student), wishedCmp);

```

Derefter starter et mindre loop, som fortsætter indtil at den støder på den første studerende der er blevet ønsket, på grund af programmets opsætning, starter vi med at sætte de personer der er blevet ønsket færrest gange (bortset fra elever der er blevet ønsket 0 gange).

Kodeudsnit 3.15: startloop løkke

```

56 |     for(i = 0; i < numOfStudents; i++)
57 |     {
58 |         if(studentList[i].wishedAmount == 0)
59 |         {
60 |             loopStart++;
61 |         }
62 |     }

```

Derefter bliver de elever med færrest personer der har ønsket dem placeret i en gruppe, og dermed er der én elev per gruppe, de bliver placeret i grupperne med en for løkke. Dette

betragtes som første stadie i funktionen, som nu er afsluttet.

Det andet stadie er at finde en partner til den første elev i hver gruppe. Siden de, der allerede er blevet placeret i en gruppe, er blevet ønsket mindst, er der ikke mange at vælge imellem, men man er trods alt, dermed sikre at de personer som ikke er blevet ønsket af specielt mange, får mindst en makker som de kan være sammen med. At finde en partner bliver gjort på 3 forskellige måder.

1. Først tjek ved hjælp af for løkker, om der er nogen elev som de har tilfælles med i ønske. Det vil sige en som de selv ønsker, men også en person som har ønsket dem.

Kodeudsnit 3.16: mutualWish løkke

```

74  /* Check if they have a mutual wish, meaning if they are wanted by a
    person that they also wish to be in a group with */
75  for(i = 0; i < groupsCount; i++)
76  {
77      for(j = 0; j < 3; j++)
78      {
79          for(k = 0; k < numOfStudents; k++)
80          {
81              if(!studentList[k].isInGroup)
82              {
83                  if(strcmp(studentList[i].name, group[0][1].notWant) ==
                        0)
84                  {
85                      isWanted = false;
86                  }
87                  if(strcmp(studentList[k].name, group[i][0].doWant[j])
                        == 0 && strcmp(group[i][0].name, studentList[k].
                        doWant[j]) == 0 && groupSizes[i] < 2 && isWanted)
88                  {
89                      group[i][1] = studentList[k];
90                      studentList[k].isInGroup = true;
91                      groupSizes[i]++;
92                  }
93              }
94          }
95      }
96  }

```

2. Hvis nogle grupper ikke fandt nogen tilfælles, skal der derefter kigges på deres ønsker,

og se om nogle af personernes ønsker stadigvæk er ledige.

Kodeudsnit 3.17: deres ønsker løkke

```

98      /* if a person cant be matched with a mutural wish, they get a wish
      instead */
99  isWanted = true;
100
101  for(i = 0; i < groupsCount; i++)
102  {
103      if(groupSizes[i] < 2)
104      {
105          for(j = 0; j < 3; j++)
106          {
107              for(k = 0; k < numOfStudents; k++)
108              {
109                  if (!studentList[k].isInGroup && strcmp(studentList
110                  [k].name, group[i][0].doWant[j]) == 0 &&
111                  groupSizes[i] < 2)
112                  {
113                      group[i][1] = studentList[k];
114                      studentList[k].isInGroup = true;
115                      groupSizes[i]++;
116                  }
117              }
118          }
119      }
120  }

```

3. Hvis alle personer fra deres ønskeliste er blevet lagt i en gruppe, bliver der kigget på hvem de er ønsket af

Kodeudsnit 3.18: var Ønsket løkke løkke

```

121      /* If group is still less than 2, see who theyre wished by instead
      */
122  for(i = 0; i < groupsCount; i++)
123  {
124      if(groupSizes[i] < 2)
125      {
126          for(j = 0; j < group[i][0].wishedAmount; j++)
127          {
128              for(k = 0; k < numOfStudents; k++)
129              {
130                  if(strcmp(studentList[k].name, group[i][0].notWant) ==
131                  0)

```

```

131         {
132             isWanted = false;
133         }
134         if(!studentList[k].isInGroup && strcmp(group[i][0].
            wishedBy[j], studentList[k].name) == 0 &&
            groupSizes[i] < 2 && isWanted)
135         {
136             group[i][1] = studentList[k];
137             studentList[k].isInGroup = true;
138             groupSizes[i]++;
139         }
140     }
141     isWanted = true;
142 }
143 }
144 }

```

4. Hvis der på dette tidspunkt stadig er grupper med kun en person i, bliver der tildelt en person fra studentList som de ikke specifict har fravalgt, altså valgt som "notWant"

Kodeudsnit 3.19: tilføj noWishes løkke

```

147         /* If there still is a person without a first partner, add someone
            that werent wished */
148     for(i = 0; i < groupsCount; i++)
149     {
150         if(groupSizes[i] < 2)
151         {
152             for(j = 0; j < numOfStudents; j++)
153             {
154                 if(strcmp(studentList[k].name, group[i][0].notWant) == 0)
155                 {
156                     isWanted = false;
157                 }
158
159                 if(!studentList[j].isInGroup && studentList[j].wishedAmount
                    == 0 && groupSizes[i] < 2 && isWanted)
160                 {
161                     group[i][1] = studentList[j];
162                     studentList[j].isInGroup = true;
163                 }
164             }
165             isWanted = true;
166         }
167     }

```

Dermed burde alle have en makker per gruppe. Og dette kan så hjælpe med det næste stadie af funktionen.

Grunden til at der bliver dannet par er ikke kun for at sikre at dem med lavest mængde gange de er ønsket får sig en partner. Men også for at bruge flere personer som data til at tilføje resten af eleverne til en gruppe. Dette er den største del af funktionen, som tager sig af en stor mængde af arbejdet. For at finde ud af hvor de forskellige personer passer bedst, laver den et lille form for pointsystem til at finde ud af hvor personen passer bedst, løkken kører overordnet på iterationer, Det vil sige at der kun bliver tilføjet en person per loop af gangen, mest for at sikre sig af der ikke ender grupper med for eksempel 6 personer, og en anden gruppe kun med 2. På denne måde bliver der delt personer op lige fordelt. Hver gang kører løkken igennem en studerende, denne løkke er nested under iterationer, siden at løkken skal starte igen efter den iteration er slut. Hele dette stykke er en række af nested for løkker, på samme måde som i forrige stadie. Løkken bruger `strcmp` til at sammenligne om de studerende i gruppens ønsker er ens med den studerendes navn. Hvis der er et match får den studerende et point, og det samme den anden vej rundt. Det vil sige at hvis den studerendes ønsker er ens med elevens navn i det givne gruppe.

Kodeudsnit 3.20: Den pointgivende løkke

```

169     int predictedGroup = -1, highestPoints = 0, currentPoints = 0;
170
171     /* add the rest of the students to a group */
172     for(iteration = 3; iteration < maxMembers; iteration++)
173     {
174         for (i = 0; i < numOfStudents; i++)
175         {
176             if (!studentList[i].isInGroup && studentList[i].wishedAmount > 0)
177             {
178                 for (j = 0; j < groupsCount; j++)
179                 {
180                     if (groupSizes[j] < iteration)
181                     {
182                         for (k = 0; k < groupSizes[j]; k++)
183                         {
184                             for (l = 0; l < 3; l++) {
185                                 if (strcmp(studentList[i].doWant[l], group[j]
186                                     ][k].name) == 0)
187                                     {
188                                         currentPoints++;

```

```

188         }
189     }
190 }
191
192     if (groupSizes[j] < iteration && currentPoints >
        highestPoints)
193     {
194         highestPoints = currentPoints;
195         predictedGroup = j;
196     }
197
198 }
199
200 }
```

Efter hver gang at løkken har fuldenet løkken igennem alle elever i en gruppe, sammenligner den med den integer der holder hvor mange point der maks er blevet givet fra en gruppe (highestPoints). Hvis den givne gruppe har givet flest point, altså større end highestPoints. Vil predictedGroup (som holder hvilken gruppe der indtil videre har givet flest point i udbytte) blive skiftet til at være j, som er den gruppe der er blevet loopet igennem. Derefter bliver nuværende point (currentPoints) sat til 0 igen.

Kodeudsnit 3.21: Placering af elev i gruppe

```

202     if (predictedGroup != -1) {
203         group[predictedGroup][iteration - 1] = studentList[i];
204         studentList[i].isInGroup = true;
205         groupSizes[predictedGroup]++;
206     }
207     predictedGroup = -1;
208     highestPoints = 0;
209 }
210 }
211 }
```

Dette bliver gjort igennem alle grupper som ikke har fået et medlem endnu igennem den nuværende iteration. Denne løkke stopper når der iterationen er nået til max antal medlemmer – 1, siden nogle af grupperne kommer potentielt til at have en ekstra i gruppen, alt efter hvor mange elever der er indsat i input filen.

Tilbage har vi dem som slet ikke blev ønsket. De skal selvfølgelig helst i en gruppe med nogen som de har ønsket. Siden vores primære mål er at finde en gruppe baseret på ønsker.

Starter vi derefter en for løkke af samme design som de tidligere, på samme måde med et pointsystem. Igen bliver de ikke sat i en gruppe hvis gruppen allerede har antal medlemmer der svarer til maks antal medlemmer – 1.

Kodeudsnit 3.22: Placering af elev der ikke blev ønsket i gruppe

```

213      /*Take care of the people without any person that wished for them by
        first checking who they wished for, again with a point assignment
        system*/
214      currentPoints = 0;
215      predictedGroup = -1;
216      highestPoints = 0;
217
218      for(i = 0; i < numOfStudents; i++)
219      {
220          if(!studentList[i].isInGroup)
221          {
222              for(j = 0; j < groupsCount; j++)
223              {
224                  if(groupSizes[j] == maxMembers - 2)
225                  {
226                      for(k = 0; k < maxMembers - 1; k++)
227                      {
228                          for(l = 0; l < 3; l++)
229                          {
230                              if(strcmp(studentList[i].doWant[l], group[j][
                                  k].name) == 0 || strcmp(group[j][k].
                                  doWant[l], studentList[i].name) == 0)
231                                  {
232                                      currentPoints++;
233                                  }
234                              }
235                          }
236                      }
237                      if(currentPoints != 0 && currentPoints > highestPoints)
238                      {
239                          highestPoints = currentPoints;
240                          predictedGroup = j;
241                      }
242                      currentPoints = 0;
243                  }
244
245          if(highestPoints != 0)

```

```

246         {
247             group[predictedGroup][maxMembers - 2] = studentList[i];
248             studentList[i].isInGroup = true;
249             groupSizes[predictedGroup]++;
250
251         }
252         highestPoints = 0;
253         predictedGroup = -1;
254     }
255 }

```

Til sidst har vi potentielt nogle elever som stadig ikke er kommet i en gruppe baseret på de tidligere metoder og løkker. Disse elever bruger vi den sekundære metode til at finde om der er en gruppe som har et gennemsnits ambitionsniveau som svarer til deres eget ambitionsniveau, og en gruppe som ikke har en elev der ikke vil have den elev i gruppen. For at gøre dette har vi dermed en for løkke som beregner det gennemsnitlige ambitionsniveau i hver gruppe.

Kodeudsnit 3.23: Beregning af ambitionsniveau

```

257
258     /* Calculate average Ambitionlevel of every group */
259     int total = 0;
260     int avg = 0;
261
262     for(i = 0; i < groupsCount; i++)
263     {
264         for(j = 0; j < groupSizes[i]; j++)
265         {
266             total += group[i][j].ambitionLevel;
267         }
268         avg = total / groupSizes[i];
269         avgAmbition[i] = avg;
270         avg = 0;
271         total = 0;
272     }

```

Derefter, på samme måde som tidligere, kører vi en løkke som finder gruppen med det tætteste ambitionsniveau, og inden i gruppen har fravalgt dem. Bare i stedet for at have highestPoints, så har vi lowestDiff, altså "Lowest difference". Og efter alle grupper er loopet igennem, bliver de indsat på den gruppe. Og størrelsen på gruppen bliver større, dermed bliver groupSizes[gruppenummer] plusset med en.

Kodeudsnit 3.24: Find Gruppe med gennemsnitlig ambitionsniveau

```

273      /*Take care of the remainder, meaning the groups with one extra
        person in it by looking at ambition level*/
274      int lowestDiff = 99999;
275      int currDiff = 0;
276      int lowestGroup = -1;
277
278      for(i = 0; i < numOfStudents; i++)
279      {
280          if(!studentList[i].isInGroup)
281          {
282              for(j = 0; j < groupsCount; j++)
283              {
284                  for(k = 0; k < groupSizes[j]; k++)
285                  {
286                      if(strcmp(studentList[i].name, group[j][k].notWant)
287                         == 0)
288                      {
289                          isWanted = false;
290                      }
291                      if (groupSizes[j] == maxMembers - 1 && !studentList[i].
292                         isInGroup)
293                      {
294                          if (studentList[i].ambitionLevel > avgAmbition[j])
295                          {
296                              currDiff = studentList[i].ambitionLevel -
297                                  avgAmbition[j];
298                          }
299                          else if (studentList[i].ambitionLevel < avgAmbition[j]
300                             ])
301                          {
302                              currDiff = avgAmbition[j] - studentList[i].
303                                  ambitionLevel;
304                          }
305                      }
306                  }
307              }
308              if (currDiff < lowestDiff && isWanted) {
309                  lowestDiff = currDiff;
310                  lowestGroup = j;
311              }
312          }
313      }

```

```

308         isWanted = true;
309
310         studentList[i].isInGroup = true;
311         group[lowestGroup][groupSizes[lowestGroup]] = studentList[i];
312         groupSizes[lowestGroup]++;
313     }
314
315 }
```

Funktionen returner ikke noget (void). Siden group pointeren er den som bliver kigget på når programmet danner output filen.

3.3.5 Output

Når sortBelbin eller sortWishes, har udfyldt groups arrayet dannes der en output fil, som indeholder de færdigt dannede grupper.

Kodeudsnit 3.25: Output

```

30 FILE *outFP = freopen("output.txt", "w", stdout);
```

Som det ses i kodeudsnit [3.25](#) laver vi en ny file pointer outFP i main.c, som har parametrene output.txt, som er filens navn, w som betyder at det er write-mode på en ny tom fil, og til sidst stdout, som gør at alt vi skriver i standard output, som printf gør, skriver i filen i stedet for i terminalen.

Funktionen printGroups er selve funktionen hvor grupperne bliver udskrevet.

Kodeudsnit 3.26: outputPrint

```

73 void printGroups(student **groups, const int groupAmount, const int
    maxStudent)
74 {
75     int i,j,k;
76     char roleStrBuffer[3][4];
77     for(i = 0; i < groupAmount; i++)
78     {
79         int n = studentsInGroup(groups[i],maxStudent);
80         printf("GRUPPE %d\n",i+1);
81
82         for(j = 0; j < n;j++)
83         {
84             for(k = 0; k < MAX_ROLES; k++)
```

```

85 |         {
86 |             roleToStr(groups[i][j].roles[k], roleStrBuffer[k]);
87 |         }
88 |         printf("%-20s (%d, %s, %s, %s, %-20s, %-20s, %-20s, %-20s)\n",
89 |             groups[i][j].name, groups[i][j].ambitionLevel,
90 |             roleStrBuffer[0], roleStrBuffer[1], roleStrBuffer[2],
91 |             groups[i][j].doWant[0], groups[i][j].doWant[1],
92 |             groups[i][j].doWant[2], groups[i][j].notWant);
93 |     }
94 |     printf("\n");
95 |
96 | }
97 | }

```

I det første *for-loop* på linje 77 i `groups.c`, kodeudsnit 3.26, bliver int `n` sat til antallet af medlemmer i gruppen (linje 79), og i næste linje bliver der udskrevet gruppenummeret. Dette *for-loop* kører altså igennem indtil at der ikke er flere grupper.

Inde i det *for-loop* på linje 82, kører vi igennem for hver person der er i gruppen, og udskriver navn, ambitionsniveauet, Belbin roller, 3 ønsker, og 1 der ikke ønskes.

For at udskrive Belbins roller for hver person, kører vi igennem endnu et *for-loop* (linje 84), som bruger funktionen `roleToStr` for hver rolle, som er et int array, og "konvertere" det til et 2D-char array (`roleStrBuffer`)

Formatet som programmet udskriver output filen kan ses under afsnit 3.4 Test af programmet.

3.4 Test af programmet

Hvis programmet køres uden en input-fil, vil programmet automatisk lave en ny input-fil i samme mappe, som programmet ligger i. Denne input-fil vil indeholde vejledning til, hvordan tekstfilen skal bygges op og giver en plads i bunden, hvor dataen skal indtastes, se bilag F. Til test af programmet har vi indtastet dummy-data på 26 nuværende universitetsstuderende, baseret på et dokument udarbejdet af fem grupper på Aalborg universitet. I dette dokument har alle grupperne, som arbejder med problemet "Gruppedannelse i Gymnasiet", indtastet deres Belbin-grupperollerresultater. Disse data er blevet skrevet ind i vores input-fil samtidig med, at vi har angivet et fiktivt ambitionsniveau samt fiktive ønsker og fravalg. Dette er gjort for at få en realistisk repræsentation af, hvor mange af hver grupperolle, der ville være i en klasse, så der ikke bare var præcis lige mange af hver og sørge for at programmet kunne håndtere at Belbins grupperoller ikke gik op. Se bilag G.

I vores test af programmet har vi testet både med en social og faglig sortering, hvor antallet af grupper har været fem. Gruppeantallet på fem er valgt ud fra, at det ville skabe 5-mandsgrupper, hvor der ville være én til rest. På den måde bliver programmets håndtering

af "restelever" også testet.

Med disse forudsætninger vil programmet give følgende output, hvis den er sat til den fagligesortering: Da der kun er to i vores dataeksempel, som har grupperollen kontaktskaber, er de

GRUPPE 1							
Mathias Jacobsen	(4, kon, ana, spe,	Ida Christiansen	, Taniya Henriksen	, Rikke Østergaard	, Teis Harrington)	
Lasse Jensen	(5, ide, spe, org,	Ida Christiansen	, Nina Burmester	, Tobias Iversen	, Adrian Plesner)	
Ida Christiansen	(5, koo, for, org,	Nina Burmester	, Tobias Iversen	, Leon Christensen	, Johan Krogh)	
Leon Christensen	(5, spe, org, afs,	Jens Christensen	, Oliver Sørensen	, Nina Burmester	, Mikkel Jensen)	
Rikke Østergaard	(3, koo, iga, spe,	Johan Krogh	, Adrian Plesner	, Christopher Davids	, Simon Christensen)	
GRUPPE 2							
Kaare Hattel	(1, koo, iga, kon,	Andreas Andersen	, Martin Kaldahl	, Anders Aaen	, Emil Aagren)	
Teis Harrington	(4, spe, for, ide,	Mathias Jacobsen	, Sámal Kristiansson	, Emil Aagren	, Rikke Østergaard)	
Adrian Plesner	(3, iga, spe, afs,	Morten Nielsen	, Emil Rasmussen	, Frederik Olesen	, Ida Christiansen)	
Oliver Sørensen	(5, ana, for, spe,	Lasse Jensen	, Tobias Iversen	, Nina Burmester	, Johan Krogh)	
Jens Christensen	(5, org, ana, for,	Lasse Jensen	, Oliver Sørensen	, Ida Christiansen	, Taniya Henriksen)	
GRUPPE 3							
Morten Nielsen	(3, ide, for, afs,	Emil Rasmussen	, Frederik Olesen	, Rikke Østergaard	, Leon Christensen)	
Tobias Iversen	(5, org, spe, for,	Leon Christensen	, Lasse Jensen	, Jens Christensen	, Frederik Olesen)	
Mikkel Jensen	(2, iga, ana, for,	Taniya Henriksen	, Sámal Kristiansson	, Emil Aagren	, Frederik Olesen)	
Anders Aaen	(1, iga, ana, afs,	Simon Christensen	, Anders Nørgaard	, Martin Kaldahl	, Johan Krogh)	
Martin Kaldahl	(1, org, iga, spe,	Philip Michaelsen	, Kaare Hattel	, Anders Nørgaard	, Taniya Henriksen)	
GRUPPE 4							
Johan Krogh	(3, for, ide, spe,	Rikke Østergaard	, Frederik Olesen	, Christopher Davids	, Nina Burmester)	
Nina Burmester	(5, for, spe, org,	Ida Christiansen	, Lasse Jensen	, Oliver Sørensen	, Sámal Kristiansson)	
Christopher Davids	(3, for, afs, spe,	Rikke Østergaard	, Johan Krogh	, Emil Rasmussen	, Tobias Iversen)	
Sámal Kristiansson	(2, org, iga, ana,	Emil Aagren	, Mikkel Jensen	, Taniya Henriksen	, Lasse Jensen)	
Andreas Andersen	(1, for, spe, org,	Simon Christensen	, Kaare Hattel	, Philip Michaelsen	, Mikkel Jensen)	
GRUPPE 5							
Frederik Olesen	(3, for, spe, org,	Morten Nielsen	, Christopher Davids	, Adrian Plesner	, Martin Kaldahl)	
Emil Rasmussen	(3, org, afs, spe,	Christopher Davids	, Morten Nielsen	, Frederik Olesen	, Philip Michaelsen)	
Taniya Henriksen	(2, org, for, spe,	Sámal Kristiansson	, Emil Aagren	, Mikkel Jensen	, Teis Harrington)	
Emil Aagren	(2, org, for, spe,	Mikkel Jensen	, Taniya Henriksen	, Sámal Kristiansson	, Mathias Jacobsen)	
Philip Michaelsen	(1, org, ana, afs,	Anders Nørgaard	, Anders Aaen	, Simon Christensen	, Emil Rasmussen)	
Simon Christensen	(1, org, koo, iga,	Kaare Hattel	, Andreas Andersen	, Martin Kaldahl	, Morten Nielsen)	

Figur 3.2: Fagligeresultat af input eksemplet

først blevet placeret i hver deres gruppe, så flest mulige grupper har fået en kontaktskaber i dem. Dette er lykkedes, da de to kontaktskabere er kommet i to forskellige grupper. Gruppe 1 har formået at have en gruppe med en af hver grupperolle i, mens ingen af de andre fire grupper mangler mere end to grupperoller. Belbins grupperoller har taget prioritet i programmet, så grupperne blev så balancerede som muligt efter Belbins teori. Da der ikke har været nok af alle grupperoller, samt at nogle med samme grupperoller har skulle være i gruppe sammen, har programmet dannet grupper ud fra at få så mange forskellige grupperoller som muligt i hver gruppe. Det er lykkedes programmet at danne fem grupper, som var det ønskede antal grupper, hvor alle 26 elever er placeret med en balance i grupperollerne (ser man bort fra "restelevne").

Med de samme forudsætninger vil programmet give følgende output, hvis en socialsortering er valgt: Den sociale sortering har fokuseret mest på at sørge for, at ingen kom i gruppe

GRUPPE 1			
Emil Aagren	(2, org, for, spe, Mikkel Jensen	, Taniya Henriksen	, Sámal Kristiansson , Mathias Jacobsen)
Taniya Henriksen	(2, org, for, spe, Sámal Kristiansson	, Emil Aagren	, Mikkel Jensen , Teis Harrington)
Martin Kaldahl	(1, org, iga, spe, Philip Michaelсен	, Kaare Hattel	, Anders Nørgaard , Taniya Henriksen)
Jens Christensen	(5, org, ana, for, Lasse Jensen	, Oliver Sørensen	, Ida Christiansen , Taniya Henriksen)
Anders Aaen	(1, iga, ana, afs, Simon Christensen	, Anders Nørgaard	, Martin Kaldahl , Johan Krogh)
Teis Harrington	(4, spe, for, ide, Mathias Jacobsen	, Sámal Kristiansson	, Emil Aagren , Rikke Østergaard)
GRUPPE 2			
Frederik Olesen	(3, for, spe, org, Morten Nielsen	, Christopher Davids	, Adrian Plesner , Martin Kaldahl)
Adrian Plesner	(3, iga, spe, afs, Morten Nielsen	, Emil Rasmussen	, Frederik Olesen , Ida Christiansen)
Kaare Hattel	(1, koo, iga, kon, Andreas Andersen	, Martin Kaldahl	, Anders Aaen , Emil Aagren)
Emil Rasmussen	(3, org, afs, spe, Christopher Davids	, Morten Nielsen	, Frederik Olesen , Philip Michaelсен)
Leon Christensen	(5, spe, org, afs, Jens Christensen	, Oliver Sørensen	, Nina Burmester , Mikkel Jensen)
GRUPPE 3			
Lasse Jensen	(5, ide, spe, org, Ida Christiansen	, Nina Burmester	, Tobias Iversen , Adrian Plesner)
Nina Burmester	(5, for, spe, org, Ida Christiansen	, Lasse Jensen	, Oliver Sørensen , Sámal Kristiansson)
Simon Christensen	(1, org, koo, iga, Kaare Hattel	, Andreas Andersen	, Martin Kaldahl , Morten Nielsen)
Andreas Andersen	(1, for, spe, org, Simon Christensen	, Kaare Hattel	, Philip Michaelсен , Mikkel Jensen)
Philip Michaelсен	(1, org, ana, afs, Anders Nørgaard	, Anders Aaen	, Simon Christensen , Emil Rasmussen)
GRUPPE 4			
Rikke Østergaard	(3, koo, iga, spe, Johan Krogh	, Adrian Plesner	, Christopher Davids , Simon Christensen)
Johan Krogh	(3, for, ide, spe, Rikke Østergaard	, Frederik Olesen	, Christopher Davids , Nina Burmester)
Christopher Davids	(3, for, afs, spe, Rikke Østergaard	, Johan Krogh	, Emil Rasmussen , Tobias Iversen)
Morten Nielsen	(3, ide, for, afs, Emil Rasmussen	, Frederik Olesen	, Rikke Østergaard , Leon Christensen)
Mathias Jacobsen	(4, kon, ana, spe, Ida Christiansen	, Taniya Henriksen	, Rikke Østergaard , Teis Harrington)
GRUPPE 5			
Ida Christiansen	(5, koo, for, org, Nina Burmester	, Tobias Iversen	, Leon Christensen , Johan Krogh)
Tobias Iversen	(5, org, spe, for, Leon Christensen	, Lasse Jensen	, Jens Christensen , Frederik Olesen)
Mikkel Jensen	(2, iga, ana, for, Taniya Henriksen	, Sámal Kristiansson	, Emil Aagren , Frederik Olesen)
Oliver Sørensen	(5, ana, for, spe, Lasse Jensen	, Tobias Iversen	, Nina Burmester , Johan Krogh)
Sámal Kristiansson	(2, org, iga, ana, Emil Aagren	, Mikkel Jensen	, Taniya Henriksen , Lasse Jensen)

Figur 3.3: Socialresultat af input eksemplet

med den, som de har fravalgt. Dette er lykkedes i vores eksempel. På trods af, at Teis Harrington har ønsket Mathias Jacobsen, så er de ikke endt i gruppe sammen, da Mathias har fravalgt Teis. Programmet fokuserer først på at danne par af dem, som har ønsket hinanden, som herefter placeres i grupper. Derefter placerer den andre i grupperne, ud fra om de er ønsket af en eller har ønsket en i gruppen. Dette kan ikke altid lade sig gøre, derfor bliver nogle fordelt efter ambitionsniveauet i gruppen. Der er dog to i gruppe 2, som ikke har ønsket nogen i gruppen og ikke er blevet ønsket af nogen i gruppen, der er her taget udgangspunkt i deres ambitionsniveau, som har balanceret gruppen, så deres ambitionsniveau passer. Også her inddeler programmet alle 26 elever i fire 5-mandsgrupper og en 6-mandsgruppe. Programmet håndterer derved "restelevne", som den skal, og derved

4. Evaluering og konklusion

4.1 Diskussion

Det udviklede program forsøger at danne de bedst mulige grupper af elever i en given klasse ud fra det valgte fokus: enten socialt eller fagligt, der bygger på elevens ambitionsniveau og henholdsvis elevernes egne ønsker og Belbins teori.

Hensigten med programmet er dog ikke, at nye grupper skal kunne dannes ud fra samme data, eftersom programmet følger en deterministisk model.

Da programmet netop prøver at finde den bedst mulige sammensætning, vil den aldrig give forskellige gruppesammensætninger - bortset fra forskellen mellem de to sorteringsmuligheder - så længe input forbliver uændret.

For at programmet kan danne nye grupper, kræver det, at noget af dataen har ændret sig. Det kan måske lade sig gøre at bruge programmet igen i andre fag, eftersom elever kan have et andet ambitionsniveau. Udover det anderledes ambitionsniveau har eleverne muligvis også en afvigelse fra første gang programmet bruges, i hvem de ønsker at være eller ikke at være, i gruppe med.

Der er patent på Belbins grupperolletest, gymnasier skal derfor betale for at gøre brug af testen, samt sætte lærere på kurser, så de kan afholde testen korrekt. At afsætte ressourcer til kursusgange af lærere hos autoriseret grupperoller, fra f.eks. Belbin teori er ikke tilgængeligt på de danske gymnasier. Samt, er det ikke mange, hvis overhovedet nogle, danske uddannelsessteder gør brug af gruppeteorier som Belbins. [26] Dette gør, at vores faglige gruppesorteringsmetode muligvis ikke er særlig brugbar i virkeligheden.

4.2 Konklusion

Vi har igennem rapporten undersøgt forskellige gruppedannelsesteorier og endte med at fokusere på de socialpsykologiske teorier Belbins grupperoller og Myers-Briggs personlighedstyper. Myers-Briggs endte med ikke at blive brugt i vores løsning, da vi ikke fandt den forenlig med Belbins teori, fordi der ikke var nok research på, hvilke Myers-Briggs-typer ville være gode sammen i en gruppe. I vores løsning har vi desuden lagt vægt på gymnasielærers egne udtalelser og erfaringer med at blande grupper. Der bliver dannet et billede af, at de bedste grupper er baseret på elevens egne ønsker, da eleverne er mest tilfredse på den måde. Eleverne føler bl.a., at de har haft indflydelse på gruppedannelsen, hvilket er vigtigt i forhold

til at danne en velfungerende og tilfreds gruppe.

Ud fra bogen *"Når gymnasiet er en fremmed verden"* fandt vi ud af, hvilke metoder der blev brugt til at danne grupper i gymnasierne, og hvad der var godt og dårligt. Da bogen er 10 år gammel, lavede vi et spørgeskema, hvor det primære formålet var at finde elevernes holdning, og hvilke metoder til hvordan grupper bliver lavet nu. Det ses ud fra spørgeskemaet at der er en lighed til bogens interviews, da eleverne primært svarede selvvalgte og lærerbestemte grupper. Det var dog svært at få en definitiv konklusion ud fra spørgeskemaet, da der ikke blev sendt ud til nok gymnasier, for at få et repræsentativt resultat.

Ud fra vores analyser og undersøgelser er det endt ud i et værktøj til lærere, som kan hjælpe med at gøre gruppedannelse hurtigere og nemmere. Programmet har to forskellige sorteringsmetoder: En faglig gruppesortering og en social gruppesortering. Den faglige sortering tager udgangspunkt i Belbins grupperolleteori og elevernes eget ambitionsniveau. Dette er gjort på baggrund af, at Belbins grupperolleteori handler om, hvordan man sammensætter en gruppe, så den bedst kan arbejde sammen, således alle ansvarsområderne er dækkede. Den sociale sortering tager udgangspunkt i elevernes egne ønsker, da gymnasielærere og elever har udtrykt, at grupper får det bedste faglige udbytte, når eleverne har det socialt godt i grupperne. Dette værktøj hjælper lærere, så de ikke selv behøver at have et overblik over alle elevers ønsker eller grupperoller for at danne grupper, men blot kan få eleverne til at sende informationerne. Dette gør lærernes arbejde nemmere og hurtigere, da læren ikke skal danne sig et overblik og derefter tage alle ønsker og grupperoller i betragtning, når de skal danne grupperne. Værktøjet indsætter alle de indtastede elever i, hvad programmet anser for at være, de bedste grupper, enten for at opnå de bedste faglige resultater med Belbin eller for at afhjælpe konflikterne på bedst mulig måde ved hjælp af elevernes egne ønsker.

4.3 Forbedring af programmet

Vi vil her diskutere programmet, og hvordan det kan videreudvikles og optimeres. Der vil blive præsenteret forbedrings- og optimeringsforslag, som ikke nødvendigvis er afhængige af hinanden.

Brug til andre uddannelser

Programmet ville igennem små justeringer kunne bruges til andre uddannelser. Den vil kunne bruges hos udskoling i folkeskolen uden store justeringer, dog kan ambitionsniveau blive et problem, da folkeskolen er en tvunget skole, så ambitionsniveauerne kan være mere spredte og sværere for lærerne at få seriøse svar fra eleverne om, hvor de ligger ambitionsmæssigt. Vores skala kan være for lille i forhold til det mere spredte ambitionsniveau i folkeskolen.

En justering til dette ville være at justere vores skala til ambitionsniveau, så den f.eks. gik 1-10 i stedet for 1-5. Folkeskole elevers grupperoller, kan måske også være besværligt at arbejde med, da elever i disse aldre er under personligheds udvikling, hvilket kan betyde deres grupperoller ikke er faste, og derved er vores determinitiske løsning, måske ikke altid hensigtsmæssig. Derfor ville det være mest fordelagtigt at benytte den sociale gruppe fokus til gruppedannelse på udskoling i folkeskoler.

Det ville kræve større justeringer, før programmet ville passe til universiteterne, fordi det ville være svært at få alle de studerende til at tage en belbintest før uddannelsesstart. så derfor ville det være urealistisk at danne de første grupper ud fra den fagligesortering. Den sociale sortering ville heller ikke kunne bruges, da de studerede ikke kender hinanden endnu. Programmet ville i stedet være mere optimalt at bruge til senere semestre, da studerende kender hinanden, og en fuldstændig indsamling af samtlige individers grupperoller (fra Belbins grupperolletest) ville også være mere realistisk når de studerende er faldet til.

Ikke deterministisk

En videreudvikling af programmet kunne være at gøre programmet ikke deterministisk, så den med den samme gymnasieklasse ville kunne skabe flere gruppekombinationer. Dette skulle implementeres med en tilfældighedsfaktor i programmet.

Socialegrupper optimering

Vores nuværende program ville ikke danne en gruppe af 4, som allesammen har ønsket hinanden. Dette skyldes at programmet starter med at danne par ud fra ønskerne og fylder derefter disse grupper op med andre, som endnu ikke har en partner, og har ønsket en, eller er blevet ønsket af en fra gruppen. Dette kunne justeres, så den ikke danner par og sætter dem i grupper, men at den først danner grupperne, når alle er sat sammen i en gruppe, så dem som er sat sammen som par stadig kan sættes sammen med et andet par.

Bibliografi

- [1] Anne Boie Johannesson. *Grupper – hvordan inddeler vi?* <https://gymnasieskolen.dk/grupper--hvordan-inddeler-vi>, 2015.
- [2] Mark Huxham & Ray Land. „Innovations in Education and Training International Volume 37“. I: *Assigning Students in Group Work Projects. Can We Do Better than Random?* 2000.
- [3] Aase Bitsch Ebbensgaard Lars Ulriksen Susanne Murning. *Når gymnasiet er en fremmed verden. Eleverfaringer – social baggrund – fagligt udbytte*. 1. ed. © Samfundslitteratur, 2009, 2009.
- [4] Penelope L. Peterson et al. *The Social Context of Instruction. Group Organization and Group Processes.* <https://files.eric.ed.gov/fulltext/ED268075.pdf>, 1984.
- [5] David W Johnson og Roger T Johnson. *What is cooperative learning?* <http://www.cooperation.org/what-is-cooperative-learning/>, 2012.
- [6] Undervisnings Ministeriet. *Belbins roller i gruppearbejdet (for gymnasieelever)*. <https://www.emu.dk/modul/belbins-roller-i-gruppearbejdet-gymnasieelever>, 2016.
- [7] Belbin UK. *Belbin for students*. <https://www.belbin.com/media/1336/belbin-for-students.pdf>, 2015.
- [8] Niels Brøgger Christian Romby Poulsen Anette Dalskov Niels Gram Bentsen Jens Kristian Sommer. „Grupper og team“. I: *Organisation*. Systime, 2015.
- [9] Belbin group. *The nine Belbin team roles*. <https://www.belbin.com/about/belbin-team-roles/>.
- [10] Team Technology. *Myers Briggs Personality Types*. <https://www.teamtechnology.co.uk/tt/t-articl/mb-simpl.htm>, 2018.
- [11] NERIS Analytics Limited. *16Personalities*. <https://www.16personalities.com/da/personlighedstyper>, 2018.
- [12] NERIS Analytics Limited. *16Personalities*. <https://www.16personalities.com/articles/roles-analysts>, 2018.

- [13] NERIS Analytics Limited. *16Personalities*. <https://www.16personalities.com/articles/roles-diplomats>. 2018.
- [14] NERIS Analytics Limited. *16Personalities*. <https://www.16personalities.com/articles/roles-sentinels>. 2018.
- [15] NERIS Analytics Limited. *16Personalities*. <https://www.16personalities.com/articles/roles-explorers>. 2018.
- [16] Peter Hagedorn-Rasmussen (red.) og Anita Mac (red.) „Projektarbejdets Komplexitet 2. udgave“. I: Kapitel 6 Projektarbejdets frugtbare kriser. 2018.
- [17] Heather Savigny; André Munro. *Free riding*. <https://www.britannica.com/topic/free-riding>. 2016.
- [18] Dorte Frost og Malene Dietz m.fl. „Projekt Samfundsfag - Teori og metode 2 2. udgave“. I: Kapitel 6 Projektarbejdets frugtbare kriser. Jun. 2016.
- [19] MBTI®. <http://www.persontests.dk/personlighedstests/mbti/>.
- [20] Alt om ledelse. *Belbin teamroller*. <https://altomledelse.dk/belbins-teamroller-staerke-teams/>.
- [21] Lars Ulriksen & Henriette T. Holmgaard. *Projektarbejde på HTX - erfaringer og udfordringer i projektvejledning*. <https://tidsskrift.dk/mona/article/view/36594>. 2008.
- [22] A317b Spørgeskema. <https://docs.google.com/forms/d/1P3hR1-6yqaRfzHWZ3JzVt-93CKJw10Rd6s4MSLeYMfg/edit#responses>.
- [23] EMU. *Belbins roller i gruppearbejdet (For gymnasieelever)*. <https://www.emu.dk/modul/belbins-roller-i-gruppearbejdet-gymnasieelever>.
- [24] Lars Ulriksen & Henriette T. Holmgaard. *Projektarbejde på HTX - erfaringer og udfordringer i projektvejledning*. <http://potential.dk/belbins-9-teamroller/belbin-cases/belbin-og-dtu-studiegrupper/>. 2008.
- [25] a317b. *sortByWishFlowdiagram*. <https://i.imgur.com/QxUSYda.png>. 2018.
- [26] Potential.dk og Meredith Belbin. *Belbins 9 teamroller*. <http://potential.dk/belbins-9-teamroller/>.

A. Samtale med Simon Søndergaard Christensen

Simon Søndergaard Christensen

Aktiv for 1 t. siden

25. OKTOBER 12:30

Hey. Jeg har faktisk et muligt sociologisk spørgsmål.

Vi har fået feedback på hvad vi har skrevet indtil videre. Vi skal definere hvad der menes med negativ og positiv gruppedannelse.

Så hvordan definere man ordene negativ og positiv i gruppedannelses sammenhæng?

Den driller nemlig mig lidt 😊

25. OKTOBER 13:48

Mit bedste bud er:

Negativ: Hvis gruppemedlemmerne føler at de ikke har haft tilstrækkelig indflydelse på gruppedannelsesprocessen, og derfor føler afmagt, frustration eller lign.

Positiv - hvis gruppemedlemmer føler at de har haft tilstrækkelig indflydelse på gruppedannelsesprocessen, og derfor acceptere udfaldet af denne.

Det handler altså om hvorvidt man føler det har været en fair proces.



Noget i den stil 😊😊

B. Samtale med Nanna Louise Ben-Or

ON. 15.31

Hej Nanna.
Vi sidder og arbejder færdigt med vores Projekt om
gruppe dannelse i gymnasiet.
Tænkte du det var en god idé at benytte
Ambitionsniveau som en betingelse for
velfungerende grupper?

TO. 06.36



Ja, absolut. Det kan variere fra dag til dag, og fra fag
til fag. Så det vil give dynamik, og nogle vil sikkert
flytte sig. De sociale relationer eleverne imellem vil
måske spænde ben for den type gruppedannelse, da
de vil sammen med vennerne, så der er brug for
lærerstyring. Det er der nu generelt 😊

C. Input eksempel

Figur C.1: Input.txt

```

#=====
# Ambitionsniveau angives på skala fra 1 til 5
#=====
# 1: Meget lavt
# 2: Lavt
# 3: Middel
# 4: Højt
# 5: Meget højt
#=====
# Grupperoller angives som de første tre bogstaver
#=====
# Handletyper | Igangsætter: iga
#              | Organisator:  org
#              | Afslutter:  afs
# Tænketyper  | Idéskaber:  ide
#              | Analysator:  ana
#              | Specialist: spe
# Sociale typer | Kontaktskaber: kon
#              | Koordinator: koo
#              | Formidler:  for
#=====
# Hvordan skal grupperne dannes?
#=====
# Antal grupper der skal dannes:
# Grupper: [ 5 ]
#
# Gruppernes fokus (SÆT ÉT KRYDS (x)):
# Fagligt: [ ]
# Socialt: [ x ]
#=====
# Elevdata skrives i bunden af dokumentet på følgende form
#=====
Navn, Ambitionsniveau, Rolle 1, Rolle 2, Rolle 3, Ønske 1, Ønske 2, Ønske 3, Fravalg.
# F.eks.:
Peter Jensen, 2, iga, ide, kon, , , , Anne Nielsen.
Anne Nielsen, 4, org, ana, koo, Peter Jensen, , , .
...
#=====
Peter, 2, iga, ide, kon, Henrik, Sophie, Arne, Anne.
Anne, 4, org, ana, koo, Peter, Lotte, Sophie, Klara.

```

D. Komplet kodeudsnit af sortBelbin funktionen

Kodeudsnit D.1: sortBelbin

```

1 void sortBelbin(student studentList[], int rolesCount[9][2], const int
  numberOfStudents, const int groupAmount, student **groups)
2 {
3     int i,j,k, studentPerGroup = numberOfStudents/groupAmount;
4
5     qsort(rolesCount,9,2*sizeof(int),rolesCmp);
6     qsort(studentList, numberOfStudents, sizeof(student), ambitionCmp);
7
8
9     for (i = 0; i < 9; i++)
10    {
11        int studentIndex = 0;
12        j = 0;
13        while(j < rolesCount[i][1] && j < groupAmount && studentIndex <
          numberOfStudents)
14        {
15            if(!studentList[studentIndex].isInGroup && studentHasRole(
              rolesCount[i][0],&studentList[studentIndex]) &&
              groupMissingRole(groups[j],rolesCount[i][0],studentPerGroup))
16            {
17                addToGroup(groups[j],&studentList[studentIndex],
                  studentPerGroup);
18                j++;
19            }
20            studentIndex++;
21        }
22    }
23    for(i = 0; i < numberOfStudents; i++)
24    {
25        if(!studentList[i].isInGroup)
26        {
27            j = 0;
28            while(j < groupAmount && !studentList[i].isInGroup)
29            {

```

```
30         if(studentsInGroup(groups[j], studentPerGroup) <
31             studentPerGroup)
32         {
33             k = 0;
34             while (k < MAX_ROLES && !studentList[i].isInGroup)
35             {
36                 if(groupMissingRole(groups[j], studentList[i].roles[k],
37                     studentPerGroup))
38                 {
39                     addToGroup(groups[j], &studentList[i],
40                         numberOfStudents);
41                 }
42                 else
43                 {
44                     k++;
45                 }
46             }
47             j++;
48         }
49         if (!studentList[i].isInGroup) {
50             j = findBestGroup(&studentList[i], groups, groupAmount,
51                 studentPerGroup + 1);
52             addToGroup(groups[j], &studentList[i], studentPerGroup + 1);
53         }
```

E. Komplet kodeudsnit af sortWishes og wishedCmp funktionen

Kodeudsnit E.1: sortWishes

```
1 |
2 | #include "sortWishes.h"
3 | #include <stdlib.h>
4 | #include <stdio.h>
5 | #include <stdbool.h>
6 | #include <string.h>
7 | #include <ctype.h>
8 | #include "groups.h"
9 |
10 | void sortWishes(student studentList[], int numOfStudents, int maxGroups,
11 |     student **group)
12 | {
13 |     /* initialize all variables */
14 |     int groupsCount = 0;
15 |     int maxMembers = numOfStudents / maxGroups + 1;
16 |     if(numOfStudents % maxMembers != 0)
17 |     {
18 |         maxMembers++;
19 |     }
20 |     int loopStart = 0;
21 |     int i, j, k, l, iteration;
22 |     bool isWanted = true;
23 |
24 |     int groupSizes[maxGroups];
25 |     int avgAmbition[maxGroups];
26 |
27 |     for(i = 0; i < numOfStudents; i++)
28 |     {
29 |         studentList[i].wishedAmount = 0;
30 |         studentList[i].isInGroup = false;
31 |     }
32 |     for(i = 0; i < maxGroups; i++)
33 |     {
34 |         groupSizes[i] = 0;
```



```
35     }
36
37     /* Count the times the different students have been wished */
38     for(i = 0; i < numOfStudents; i++)
39     {
40         for(j = 0; j < numOfStudents; j++)
41         {
42             for(k = 0; k < 3; k++)
43             {
44                 if(strcmp(studentList[i].name, studentList[j].doWant[k]) ==
45                     0)
46                 {
47                     strcpy(studentList[i].wishedBy[studentList[i].
48                         wishedAmount], studentList[j].name);
49                     studentList[i].wishedAmount++;
50                 }
51             }
52         }
53         qsort(studentList, numOfStudents, sizeof(student), wishedCmp);
54
55
56         for(i = 0; i < numOfStudents; i++)
57         {
58             if(studentList[i].wishedAmount == 0)
59             {
60                 loopStart++;
61             }
62         }
63
64         /* assign a person to each group. starting from the least wished (besides
65             those that hasnt been wished for */
66         for(i = 0; i < maxGroups; i++)
67         {
68             groupSizes[i]++;
69             group[i][0] = studentList[i];
70             studentList[i].isInGroup = true;
71             groupsCount++;
72         }
73         printf("\n");
74
75         /* Check if they have a mutural wish, meaning if they are wanted by a person
76             that they also wish to be in a group with */
77         for(i = 0; i < groupsCount; i++)
78         {
```

```

77     for(j = 0; j < 3; j++)
78     {
79         for(k = 0; k < numOfStudents; k++)
80         {
81             if(!studentList[k].isInGroup)
82             {
83                 if(strcmp(studentList[i].name, group[0][1].notWant) == 0)
84                 {
85                     isWanted = false;
86                 }
87                 if(strcmp(studentList[k].name, group[i][0].doWant[j]) ==
                        0 && strcmp(group[i][0].name, studentList[k].doWant[j
                        ]) == 0 && groupSizes[i] < 2 && isWanted)
88                 {
89                     group[i][1] = studentList[k];
90                     studentList[k].isInGroup = true;
91                     groupSizes[i]++;
92                 }
93             }
94         }
95     }
96 }
97
98 /* if a person cant be matched with a mutural wish, they get a wish
   instead */
99 isWanted = true;
100
101 for(i = 0; i < groupsCount; i++)
102 {
103     if(groupSizes[i] < 2)
104     {
105         for(j = 0; j < 3; j++)
106         {
107             for(k = 0; k < numOfStudents; k++)
108             {
109                 if (!studentList[k].isInGroup && strcmp(studentList[k
                        ].name, group[i][0].doWant[j]) == 0 && groupSizes
                        [i] < 2)
110                 {
111                     group[i][1] = studentList[k];
112                     studentList[k].isInGroup = true;
113                     groupSizes[i]++;
114                 }
115             }
116         }
117     }

```

```

118     }
119
120     isWanted = true;
121     /* If group is still less than 2, see who theyre wished by instead*/
122     for(i = 0; i < groupsCount; i++)
123     {
124         if(groupSizes[i] < 2)
125         {
126             for(j = 0; j < group[i][0].wishedAmount; j++)
127             {
128                 for(k = 0; k < numOfStudents; k++)
129                 {
130                     if(strcmp(studentList[k].name, group[i][0].notWant) == 0)
131                     {
132                         isWanted = false;
133                     }
134                     if(!studentList[k].isInGroup && strcmp(group[i][0].
135                         wishedBy[j], studentList[k].name) == 0 && groupSizes[
136                         i] < 2 && isWanted)
137                     {
138                         group[i][1] = studentList[k];
139                         studentList[k].isInGroup = true;
140                         groupSizes[i]++;
141                     }
142                 }
143                 isWanted = true;
144             }
145         }
146
147         isWanted = true;
148         /* If there still is a person without a first partner, add someone that
149            werent wished */
150         for(i = 0; i < groupsCount; i++)
151         {
152             if(groupSizes[i] < 2)
153             {
154                 for(j = 0; j < numOfStudents; j++)
155                 {
156                     if(strcmp(studentList[k].name, group[i][0].notWant) == 0)
157                     {
158                         isWanted = false;
159                     }
160
161                     if(!studentList[j].isInGroup && studentList[j].wishedAmount
162                         == 0 && groupSizes[i] < 2 && isWanted)

```

```
160         {
161             group[i][1] = studentList[j];
162             studentList[j].isInGroup = true;
163         }
164     }
165     isWanted = true;
166 }
167 }
168
169 int predictedGroup = -1, highestPoints = 0, currentPoints = 0;
170
171 /* add the rest of the students to a group */
172 for(iteration = 3; iteration < maxMembers; iteration++)
173 {
174     for (i = 0; i < numOfStudents; i++)
175     {
176         if (!studentList[i].isInGroup && studentList[i].wishedAmount > 0)
177         {
178             for (j = 0; j < groupsCount; j++)
179             {
180                 if (groupSizes[j] < iteration)
181                 {
182                     for (k = 0; k < groupSizes[j]; k++)
183                     {
184                         for (l = 0; l < 3; l++) {
185                             if (strcmp(studentList[i].doWant[l], group[j]
186                                     ][k].name) == 0)
187                             {
188                                 currentPoints++;
189                             }
190                         }
191                     }
192                     if (groupSizes[j] < iteration && currentPoints >
193                         highestPoints)
194                     {
195                         highestPoints = currentPoints;
196                         predictedGroup = j;
197                     }
198                 }
199             }
200         }
201     }
202     if (predictedGroup != -1) {
203         group[predictedGroup][iteration - 1] = studentList[i];
```

```

204         studentList[i].isInGroup = true;
205         groupSizes[predictedGroup]++;
206     }
207     predictedGroup = -1;
208     highestPoints = 0;
209 }
210 }
211 }
212
213 /*Take care of the people without any person that wished for them by
    first checking who they wished for, again with a point assignment
    system*/
214 currentPoints = 0;
215 predictedGroup = -1;
216 highestPoints = 0;
217
218 for(i = 0; i < numOfStudents; i++)
219 {
220     if(!studentList[i].isInGroup)
221     {
222         for(j = 0; j < groupsCount; j++)
223         {
224             if(groupSizes[j] == maxMembers - 2)
225             {
226                 for(k = 0; k < maxMembers - 1; k++)
227                 {
228                     for(l = 0; l < 3; l++)
229                     {
230                         if(strcmp(studentList[i].doWant[l], group[j][
                             k].name) == 0 || strcmp(group[j][k].
                             doWant[l], studentList[i].name) == 0)
231                         {
232                             currentPoints++;
233                         }
234                     }
235                 }
236             }
237             if(currentPoints != 0 && currentPoints > highestPoints)
238             {
239                 highestPoints = currentPoints;
240                 predictedGroup = j;
241             }
242             currentPoints = 0;
243         }
244     }
245     if(highestPoints != 0)

```

```

246         {
247             group[predictedGroup][maxMembers - 2] = studentList[i];
248             studentList[i].isInGroup = true;
249             groupSizes[predictedGroup]++;
250
251         }
252         highestPoints = 0;
253         predictedGroup = -1;
254     }
255 }
256
257 /* Calculate average Ambitionlevel of every group */
258 int total = 0;
259 int avg = 0;
260
261 for(i = 0; i < groupsCount; i++)
262 {
263     for(j = 0; j < groupSizes[i]; j++)
264     {
265         total += group[i][j].ambitionLevel;
266     }
267     avg = total / groupSizes[i];
268     avgAmbition[i] = avg;
269     avg = 0;
270     total = 0;
271 }
272
273 /*Take care of the remainder, meaning the groups with one extra person in
it by looking at ambition level*/
274 int lowestDiff = 99999;
275 int currDiff = 0;
276 int lowestGroup = -1;
277
278 for(i = 0; i < numOfStudents; i++)
279 {
280     if(!studentList[i].isInGroup)
281     {
282         for(j = 0; j < groupsCount; j++)
283         {
284             for(k = 0; k < groupSizes[j]; k++)
285             {
286                 if(strcmp(studentList[i].name, group[j][k].notWant)
287                    == 0)
288                 {
289                     isWanted = false;

```

```

290     }
291     if (groupSizes[j] == maxMembers - 1 && !studentList[i].
        isInGroup)
292     {
293         if (studentList[i].ambitionLevel > avgAmbition[j])
294         {
295             currDiff = studentList[i].ambitionLevel -
                avgAmbition[j];
296         }
297         else if (studentList[i].ambitionLevel < avgAmbition[j]
            ])
298         {
299             currDiff = avgAmbition[j] - studentList[i].
                ambitionLevel;
300         }
301     }
302
303     if (currDiff < lowestDiff && isWanted) {
304         lowestDiff = currDiff;
305         lowestGroup = j;
306     }
307 }
308 isWanted = true;
309
310 studentList[i].isInGroup = true;
311 group[lowestGroup][groupSizes[lowestGroup]] = studentList[i];
312 groupSizes[lowestGroup]++;
313 }
314
315 }
316
317 /* Check if someone is in a group where theyre not wished by a certain
    person */
318 /* If there is the certain edge case of people not able to be in a
    different group, then so be it unfortunately */
319
320 }
321
322 /* Compare function for the qsort function for wishedBy */
323 int wishedCmp( const void *a, const void *b)
324 {
325     student *pa = (student*)a;
326     student *pb = (student*)b;
327     return (pb->wishedAmount - pa->wishedAmount);
328 }

```

F. Inputskabelon

Figur F.1: Inputskabelon

```
#=====
# Ambitionsniveau angives på skala fra 1 til 5
#=====
# 1: Meget lavt
# 2: Lavt
# 3: Middel
# 4: Højt
# 5: Meget højt
#=====
# Grupperoller angives som de første tre bogstaver (hvis intet, skriv "x")
#=====
# Handletyper---| -Igangsætter: iga
#                  | Organisator:  org
#                  | Afslutter:  afs
# Tænketyper----| -Idéskaber:  ide
#                  | Analysator:  ana
#                  | Specialist:  spe
# Sociale typer-| -Kontaktskaber: kon
#                  | Koordinator: koo
#                  | Formidler:  for
#=====
# Hvordan skal grupperne dannes?
#=====
# Antal grupper der skal dannes:
# Grupper: [  ]
#
# Gruppernes fokus (SÆT ÉT KRYDS (x)):
# Fagligt: [  ] (Roller og ambition som prioritet)
# Socialt: [  ] (Ønsker og ambition som prioritet)
#=====
# Elevdata skrives i bunden af dokumentet på følgende form
#=====
Navn, Ambitionsniveau, Rolle 1, Rolle 2, Rolle 3, Ønske 1, Ønske 2, Ønske 3, Fravalg.
# F.eks.:
Peter Jensen, 2, iga, ide, kon, Anders Bo, Rune Pedersen, Karoline Skov, Anne Nielsen.
Anne Nielsen, 4, org, ana, koo, Peter Jensen, Karoline Skov, Anders Bo, Rune Pedersen.
...
#=====|ELEVDATA HERUNDER|=====
```


G. Dataeksempel

Figur G.1: Dataeksempel

Simon Christensen, 1, org, koo, iga, Kaare Hattel, Andreas Andersen, Martin Kaldahl, Morten Nielsen.
 Kaare Hattel, 1, koo, iga, kon, Andreas Andersen, Martin Kaldahl, Anders Aaen, Emil Aagren.
 Andreas Andersen, 1, for, spe, org, Simon Christensen, Kaare Hattel, Philip Michaelsen, Mikkel Jensen.
 Martin Kaldahl, 1, org, iga, spe, Philip Michaelsen, Kaare Hattel, Anders Nørgaard, Taniya Henriksen.
 Anders Aaen, 1, iga, ana, afs, Simon Christensen, Anders Nørgaard, Martin Kaldahl, Johan Krogh.
 Philip Michaelsen, 1, org, ana, afs, Anders Nørgaard, Anders Aaen, Simon Christensen, Emil Rasmussen.
 Emil Aagren, 2, org, for, spe, Mikkel Jensen, Taniya Henriksen, Sámal Kristiansson, Mathias Jacobsen.
 Mikkel Jensen, 2, iga, ana, for, Taniya Henriksen, Sámal Kristiansson, Emil Aagren, Frederik Olesen.
 Taniya Henriksen, 2, org, for, spe, Sámal Kristiansson, Emil Aagren, Mikkel Jensen, Teis Harrington.
 Sámal Kristiansson, 2, org, iga, ana, Emil Aagren, Mikkel Jensen, Taniya Henriksen, Lasse Jensen.
 Rikke Østergaard, 3, koo, iga, spe, Johan Krogh, Adrian Plesner, Christopher Davids, Simon Christensen.
 Adrian Plesner, 3, iga, spe, afs, Morten Nielsen, Emil Rasmussen, Frederik Olesen, Ida Christiansen.
 Christopher Davids, 3, for, afs, spe, Rikke Østergaard, Johan Krogh, Emil Rasmussen, Tobias Iversen.
 Morten Nielsen, 3, ide, for, afs, Emil Rasmussen, Frederik Olesen, Rikke Østergaard, Leon Christensen.
 Emil Rasmussen, 3, org, afs, spe, Christopher Davids, Morten Nielsen, Frederik Olesen, Philip Michaelsen.
 Johan Krogh, 3, for, ide, spe, Rikke Østergaard, Frederik Olesen, Christopher Davids, Nina Burmester.
 Frederik Olesen, 3, for, spe, org, Morten Nielsen, Christopher Davids, Adrian Plesner, Martin Kaldahl.
 Teis Harrington, 4, spe, for, ide, Mathias Jacobsen, Sámal Kristiansson, Emil Aagren, Rikke Østergaard.
 Mathias Jacobsen, 4, kon, ana, spe, Ida Christiansen, Taniya Henriksen, Rikke Østergaard, Teis Harrington.
 Ida Christiansen, 5, koo, for, org, Nina Burmester, Tobias Iversen, Leon Christensen, Johan Krogh.
 Nina Burmester, 5, for, spe, org, Ida Christiansen, Lasse Jensen, Oliver Sørensen, Sámal Kristiansson.
 Tobias Iversen, 5, org, spe, for, Leon Christensen, Lasse Jensen, Jens Christensen, Frederik Olesen.
 Leon Christensen, 5, spe, org, afs, Jens Christensen, Oliver Sørensen, Nina Burmester, Mikkel Jensen.
 Jens Christensen, 5, org, ana, for, Lasse Jensen, Oliver Sørensen, Ida Christiansen, Taniya Henriksen.
 Lasse Jensen, 5, ide, spe, org, Ida Christiansen, Nina Burmester, Tobias Iversen, Adrian Plesner.
 Oliver Sørensen, 5, ana, for, spe, Lasse Jensen, Tobias Iversen, Nina Burmester, Johan Krogh.

H. Nyt P1-projektforslag

H.1 Prissammenligner for dagligvarer

Problemstilling

Sammenligner priser og tilbud fra forskellige tilbudsaviser på dagligvarer.

Som studerende har man ofte et begrænset budget og vil derfor gerne spare så meget som muligt, et at de mest oplagte steder at spare, er på dagligvarer, da der ofte er tilbud og mange butikker fører en "priskrig" om at have de billigste varer. Dog er det svært at holde styr på nuværende dagligvaretilbud i nærheden. Hvis man skal holde sig opdateret på alle tilbud, som potentielt kan være relevante for sig, kan det blive besværligt og tidskrævende at bladere mellem tilbudsaviser, eller fysisk besøge butikkerne, hver gang man skal handle, blot for at se om æblejuicen er billigst i Føtex eller SuperBrugsen. Hvis alle de bedste tilbud var samlet på et enkelt, lettilgængeligt medie, ville brugeren kunne spare tid på og måske endda penge.

Mål

At analysere markedet og afgrænse emnet til at kunne lave et program i P1-perioden, der kan finde det bedste tilbud på en bestemt vare ud fra et antal tilbudsaviser.

Eksempler på datalogiske problemstillinger

Sammenligningsalgoritmer, at kunne genkende et produkt flere steder og muligvis differentiere mellem forskellige alternativer.

Eksempler på kontekstuelle spørgsmål og problemstillinger

Hvilke problemer løser det, at man kan finde alle tilbuddene på et sted? Vil det have en positiv eller negativ påvirkning på køber og/eller sælger. Er der tid at spare for brugeren af programmet?

Se/research eventuelt: Pricerunner, personaliseret shopping, Machine learning.