Teaching Tool

Innføring av studentaktive læringsmetoder for å utvide studentenes forståelse av programmering

T.F. Brustad[1*]

[1] Institutt for Datateknologi og beregningsorienterte ingeniørfag, Fakultet for ingeniørvitenskap og teknologi, UiT Norges Arktiske Universitet, Norge

*Kontakt. E-post: tanita.f.brustad@uit.no

Abstract: Many students associate programming with writing code in a programming language, but the programming process is much more than coding. Programming is a way of thinking, decomposing, and solving problems, where there is usually no single correct answer. In teaching programming subjects, it can sometimes be difficult to get students to plan and reflect on the choices they make when coding, and they often end up choosing the first solution they find. In this project, a pilot study was conducted where two different student-active learning methods were implemented in a programming course. The methods are intended to help students understand that programming is more than coding and to teach them various parts of the programming process. The two activities used in the project are brain teasers and oral assignment, both of which encourage creativity, reflection, and problem-solving among students.

Sammendrag: Mange studenter forbinder programmering med det å skrive kode i et programmeringsspråk, men programmeringsprosessen er mye mer enn koding. Programmering er en måte å tenke, dekomponere, og løse problemer på, hvor det som regel ikke finnes ett fasitsvar. I undervisning i programmeringsfag kan det av og til være vanskelig å få studentene til å planlegge og reflektere rundt de valgene de tar når de koder, og de ender ofte opp med å velge første løsning de finner. I dette prosjektet er det gjennomført en pilotstudie hvor to ulike studentaktive læringsmetoder er implementert i et programmeringsfag. Metodene skal hjelpe studentene å forstå at programmering er mer enn å kode, og skal lærer studentene ulike deler av programmeringsprosessen. De to aktivitetene som er brukt i prosjektet er grubleoppgaver og muntlig obligatorisk oppgave, som begge oppfordrer til kreativitet, refleksjon, og problemløsning hos studentene.

Nøkkelord: Studentaktiv læring, programmering, problemløsning, grubleoppgaver, muntlig obligatorisk oppgave

DOI: 10.5324/njsteme.v9i1.6294

1 Introduksjon

På de datarelaterte utdanningene ved Institutt for datateknologi og beregningsorienterte Ingeniørfag ved UiT Norges Arktiske Universitet lærer studentene å programmere i mange fag og de har en god oversikt hva koding er. En misoppfatning blant studentene er at de ofte forbinder programmering med bare koding. Men programmering er mer enn bare evnen til å generere kode, det kan betraktes som en måte å tenke, dekomponere, og løse problemer på (Blikstein et al., 2014), hvor det som regel ikke finnes en gitt fasit. I en undervisningssituasjon har studentene, basert på min erfaring, lite refleksjoner rundt valgene de tar når de koder; de bruker ofte en prøv og feil metode. Dette kan være på grunn av tidsbegrensninger i fag, programmeringsnivå, men også fordi de ikke vet om alternativer eller "best practices". Når studentene velger en prøv og feil metode ender de ofte opp med å drive med overflatelæring ved at de velger første løsning de finner (Anderman, 1992), noe som i teorien ikke går overens med hva programmering egentlig er.

Mitt mål med dette undervisningsopplegget er å teste ut aktiviteter som skal være med på å få studentene til å tenke som en programmerer; fra ide til løsning, og vise dem at programmering er mer enn å kode. Tanken er å finne aktiviteter hvor studentene lærer på en god måte ulike deler av programmeringsprosessen.

For denne artikkelen har det blitt gjennomført en pilotstudie, på et lite antall studenter, med et formål om å:

- Finne aktiviteter som lærer studentene ulike deler av programmeringsprosessen.
- Bruke studentaktive læringsmetoder rettet mot programmering.
- Få tilbakemelding på opplevd læringsutbytte.

Gjennom dette formålet ønsker jeg at studentene skal oppnå følgende læringsutbyttet:

- L1: Forstå at programmering inneholder flere deler enn koding.
- L2: Få erfaring med hele programmeringsprosessen.
- L3: Reflektere over eget læringsutbytte når de har fått brukt hele prosessen.

Resultatet skal brukes til videreutvikling av aktiviteter for større grupper studenter på ulike nivåer.

2 Bakgrunn

Det kom tidlig indikasjoner på at dårlig planlegging i forkant øker kodefeil i større grad enn manglende kunnskap om programmeringsspråket (Soloway & Ehrlich, 1984; Soloway & Spohrer, 1988). Det er også funnet en sammenheng mellom problemløsningsevner og prestasjon på programmeringsoppgaver (Palumbo, 1990; Lishinski et al., 2016). I forskning som ser på undervisning i programmering er det ofte det å lære studentene problemløsning som er i fokus og som viser seg å være effektivt (Barnes et al., 1997; Deek et al., 1998; Carlisle et al., 2005; Aktunc, 2013; Loksa et al., 2016; Nikolic et al., 2018; Mathew et al., 2019).

Det er bevist at i STEM-fag[1] lærer studentene mest når studentaktive læringsmetoder tas i bruk, slik at studentene involveres og aktiviseres i undervisningen (Freeman et al., 2014; Wieman, 2014). Studentaktiv læring defineres som aktiviteter som involverer at studentene gjør noe og tenker over hva de gjør (Bonwell & Eison, 1991). Kjerneelementene i studentaktiv læring er, som navnet tilsier, studentaktivitet og engasjement i læringsprosessen, og er ofte motsetningen til tradisjonell forelesning hvor studentene passivt mottar informasjon (Prince, 2004). NOKUT's rapport om studentaktiv læring fra 2019 (Kantardjiev, 2019) viser klare fordeler med denne metoden i forhold til: økt interesse for faget, mer utøvende kunnskap, bedre problemløsningsevner og bedre holdninger til læring. Det er også spesifikt bevist at studentaktiv læring fungere godt for å lære programmering. I en litteraturstudie fra 2021 (Berssanette & de Francisco, 2021) med fokus på studentaktive læringsmetoder i programmering ble 38 ulike artikler gjennomgått og det ble konkludert med at metodene har en positiv innvirkning på blant annet prestasjon, motivasjon, og læringsutbytte hos studentene.

3 Løsning

I mange fag hvor programmering er hovedfokus, er det koding som får mest oppmerksomhet. Derfor valgte jeg aktivt aktiviteter til dette opplegget som fokuserte på andre deler av programmering. Barnes et al. (1997) presenterte en problemløsningssirkel med fire steg for å lære studentene programmeringsprosessen. De fire stegene var: forståelse, design, koding, og refleksjon. Denne sirkelen så ut til å ha god effekt på å lære studentene programmering, og jeg har dermed valgt å ta utgangspunkt i den når jeg har satt opp aktiviteter. Aktivitetene som ble testet innførte tre av de fire stegene i sirkelen; forståelse, design, og refleksjon. Disse ble implementert gjennom to ulike oppgavetyper; grubleoppgaver og muntlig obligatorisk oppgave (muntlig oblig), inspirert av de studentaktive læringsformene med samme navn i notatet fra UiO om studentaktive læringsformer (Tellefsen, 2018).

Ved å innføre disse aktivitetene ønsket jeg at studentene skulle oppnå læringsutbytte L1 ved at de fikk en forståelse for at det er mange valg som må tas før kodingen starter, og at de må reflektere gjennom hele prosessen, både før, underveis, og etter koding, og L2 ved at de fikk praktisk erfaring med stegene; forståelse, design, og refleksjon.

Siden faget jeg skulle bruke oppgavene i går på engelsk valgte jeg å gi oppgavene engelske navn. Navnene er valgt basert på hvilken type oppgave det er, men også basert på et navnetriks som ble presentert på et seminar kalt *The teaching trick* (Edström & Kuttenkeuler, 2020). På seminaret ble det nevnt eksempler på hvor mye navngiving kan bety for hvordan studentene oppfatter en oppgave, bl.a. forskjellen på å kalle en kompleks oppgave for homework kontra master test, og hva det hadde å si for antall klager fra studentene på kompleksiteten. Inspirert av dette bestemte jeg meg for å kalle grubleoppgavene for «Brain teasers» og muntlig oblig for «Multiple heads one tale!».

[1] STEM= Science, Technology, Engineering and Mathematics/Naturvitenskap, teknologi, ingeniørfag og matematikk.

3.1 Grubleoppgaver (Brain teasers)

Det ble satt opp to økter med grubleoppgaver, en med fokus på planlegging før programmering og en med fokus på debugging av kode. Disse knyttes opp mot forståelse og design i problemløsningssirkelen. Mine forventninger til studentene på disse oppgavene var at de skulle komme opp med så mange ideer de klarte på den tiden de hadde, og prøve å se temaet fra sider som de ikke hadde tenkt på tidligere. En økt varte 45 min, hvor studentene ble delt i grupper på 4 - 7. Gruppene fikk utdelt grubleoppgaven som først diskutertes i gruppen (30 min), og så presentertes og diskutertes i plenum (15 min). Oppgavene ble gjort tilgjengelig for studentene to dager før aktiviteten. Oppgavebeskrivelsen til begge oppgavene er presentert i Vedlegg 1.

3.2 Muntlig obligatorisk oppgave (Multiple heads one tale!)

Muntlig oblig var en aktivitet som ble gjennomført mot slutten av undervisningsperioden over et lengre tidsintervall enn grubleoppgavene. Denne aktiviteten knyttes opp mot forståelse, design, og refleksjon i problemløsningssirkelen**,** men med størst fokus på refleksjon. Mine forventninger til studentene på denne aktiviteten var at de skulle se at programmeringsoppgaver kan løses på flere ulike måter, og at tankegangen de bruker for å løse en oppgave kan være veldig forskjellig fra hvordan andre tenker. Studentene ble delt i par, og bedt om å lage en presentasjon basert på gitte spørsmål til en individuell innlevering (laboppgave) de allerede hadde gjort. Denne presentasjonen ble fremført i klassen der hvert par hadde 20 minutter presentasjonstid og 10 minutter til spørsmål og diskusjon. Etter at alle gruppene hadde presentert hadde vi en felles diskusjon på rundt 15 minutter. Gruppeinndelingen og tildelt laboppgave ble lagt ut 3 uker i forveien, mens spørsmålene kunne studentene se fra starten av emnet. Oppgaveteksten med spørsmålene som studentene skulle besvare er presentert i Vedlegg 2.

4 Analyse

Aktivitetene ble innarbeidet i et programmeringsfag på første året master. Faget består av teori og mye koding i et programmeringsspråk studentene ikke har erfaring med. Selv om dette er masterstudenter opplever vi at programmeringsferdighetene varierer i forhold til hvor studentene har tatt sin tidligere utdannelse, og at studentene fortsatt ser på programmering som koding.

Klassene jeg jobbet med var relativt få studenter i antall (8 stk.), noe som gjorde det enklere å teste ut aktivitetene, og vurdere fordeler og ulemper.

Studentene evaluerte aktivitetene ved utfylling av et anonymt evalueringsskjema (se Vedlegg 3). På den måten oppnådde studentene læringsutbytte L3 ved at de måtte reflektere over eget læringsutbyttet fra aktivitetene og evaluere aktivitetene på ulike punkter.

4.1 Grubleoppgaver

Tilbakemeldingene fra studentene var både positive og negative. De fleste studenter mente at temaet var relevant, at aktiviteten var relevant for resterende fag på utdanningen, at formatet var godt, at lengden var bra, og at aktiviteten ble godt forklart.

Studenter likte at de fikk et litt annet perspektiv på ulike deler av programmering. I forhold til læringsutbytte var studentene delt mellom at de enten følte de fikk et høyt læringsutbytte eller et lavt. Det ble kommentert at de heller ville brukt tiden til å diskutere spesifikke temaer knyttet til faget, eller at oppgavene kunne vært endret.

Mitt inntrykk av denne aktiviteten var at studentene diskuterte godt i gruppene og kom med gode svar og kreative løsninger, som de kanskje ellers ikke ville tenkt over. Jeg følte at å fokusere på ett enkelt steg i programmeringsprosessen gjorde at studentene hadde god tid til å utfolde seg kreativt i steget uten at de skynder seg for å komme til kode-steget. At studentene kom med varierte svar og ulike løsninger viser at de har ferdighetene til å utføre alle stegene i prosessen, men at de kanskje ikke forstår viktigheten av å utføre dem ordentlig. I samtale med studentene i diskusjonsdelen pratet vi om viktigheten av steget som var knyttet til oppgaven, og studentene hadde reflekterte tanker rundt temaet, som for meg viser at de lærte noe av aktiviteten.

I diskusjon med faglærer var det noen studenter som ble litt passive og kun svarte på direkte henvendelse. Om dette er fordi gruppen som diskutere blir større, eller om det er fordi faglærer er en del av diskusjonen er vanskelig å si, men så lenge studentene er aktive i gruppediskusjonen vil ikke passiviteten i denne delen, nødvendigvis, ha utslag på læringsutbyttet.

4.2 Muntlig obligatorisk oppgave

Tilbakemeldingene på muntlig oblig ga et inntrykk av at studentene syntes øvelsen var bra og gjorde at de måtte reflektere og tenke mer over valgene de hadde gjort på laboppgavene. Det ble også dratt frem at læringsutbyttet ble større av at de fikk en innsikt i medstudenter sin tankegang i forhold til implementering. Studentene var enige i at aktiviteten var relevant for faget, at læringsutbyttet var bra, og at formatet var godt. Det eneste negative som ble nevnt var at tiden brukt på muntlig oblig heller burde vært brukt på den avsluttende prosjektoppgaven i faget. Basert på dette kan det virke som at noen studenter ikke så på denne aktiviteten som en del av faget, men heller noe de måtte gjøre i tillegg.

Mitt helhetsinntrykk av denne aktiviteten var at studentene satt igjen med en større forståelse av teorien rundt oppgavene de hadde gjort, og de fikk reflektert over spørsmål som de kanskje ikke hadde tatt stilling til tidligere. Studentene kom med gode spørsmål til hverandre og de oppdaget ved flere anledninger hvor forskjellig de hadde tenkt når de implementerte laboppgaven. Jeg la også merke til at det var enklere å engasjere alle studentene til å komme med sine tanker, i motsetning til at enkelte tar over diskusjonen mens andre bare hører på.

Basert på opplevelsen i klasserommet og interaksjonen mellom studentene virket det som at læringsutbyttet i forhold til forståelse, design, og refleksjon var større for denne typen aktivitet sammenlignet med Grubleoppgaver. Årsaken kan være at studentene hadde lengre tid på denne aktiviteten og de var mer kjent med innholdet. Til gjengjeld var det mer krevende å gjennomføre muntlig oblig, som strakte seg over tre uker, i motsetning til grubleoppgaver som tok rundt 1 time å gjennomføre.

4.3 Skalering

Skaleringen av grubleoppgaver til større studentgrupper er definitivt mulig. En faglig ressurs klarer å holde styr på 10 grupper med 4-7 studenter. De parameterne som er

avhengig av skalering er oppdeling av grupper og å passe på at alle gruppene utfører aktivitetene på best mulig måte. Av disse er det det siste punktet som tar mest tid, der det må holdes oversikt over alle gruppene og diskusjonene som pågår. Det vil også være viktig å passe på at gruppene ikke blir for store, slik at alle studenter får deltatt i diskusjonen.

Skalering av muntlig oblig til større studentgrupper krever mer enn for grubleoppgaver. Den muntlige delen baserer seg på tidligere oppgaver studentene har gjort i emnet, og i tillegg skal alle studentpar presentere sitt arbeid. Det kan være mulig å kjøre parallelle sesjoner med presentasjon, og man kan da bruke samme oppgave på flere studentpar som presenterer i ulike sesjoner. For å gjøre det på denne måten vil det være nødvendig med én faglig ressurs per sesjon. Det vil også være relevant å se på muligheten for at studentene er flere enn 2 per gruppe, men gruppestørrelser på flere enn 4 kan potensielt gjøre aktiviteten mer kaotisk, både i forhold til å lage presentasjoner og høre på andres presentasjon.

5 Oppsummering

I Tabell 1 har jeg identifisert de positive og negative egenskapene til aktivitetene. De positive egenskapene til grubleoppgavene er at studentene får nye perspektiver på ulike deler av programmering, de oppdager at det kan være flere løsninger på et problem, og aktiviteten er relevant for alle programmeringsfag. De negative egenskapene er at valg av tema har stor betydning og kan være vanskelig å få riktig. I tillegg gjør formatet, med en større gruppe, det mer sannsynlig at noen studenter prater mye mer enn andre.

For muntlig oblig har jeg identifisert at de positive egenskapene er at studentene ser flere sider av samme oppgave og får innblikk i at tankegangen kan være ulik for hvordan oppgaven løses. Teorien blir forklart fra studentenes synspunkt, som gjør at studentene får en annen vinkling på stoffet, både ved å presentere for andre og ved å bli presentert for. Læringsutbyttet blir også større for de originale oppgavene ved at studentene får bearbeide oppgavene de allerede har gjennomført. De negative egenskapene er at aktiviteten er tidkrevende å gjennomføre og krever en del planlegging. I tillegg tar det bort fokus fra andre innleveringer i faget den tiden aktiviteten pågår.

Tabell 1. *Fordeler og ulemper for aktivitetene, hentet ut fra tilbakemeldinger og observasjoner.*

Grubleoppgaver	
Fordeler	**Ulemper**
• Nytt perspektiv på ulike deler av programmering. • Flere kreative løsninger kommer frem. • Relevant for alle programmeringsfag.	• Noen prater mer enn andre. • Valg av tema har stor betydning.
Muntlig oblig	
Fordeler	**Ulemper**
• Ser flere sider av samme oppgave. • Teorien blir forklart fra studentenes synspunkt. • Læringsutbytte av de originale laboppgavene blir større.	• Tidkrevende. • Tar bort fokus fra andre innleveringer i faget.

Basert på tilbakemeldingene, mine egne observasjoner, og fordelene og ulempene ved hver aktivitet har jeg sett ut tre områder som er viktig og må fungere sammen for å få mest mulig ut av aktivitetene:

- **Opplevelse:** Studentene må oppleve aktivitetene som meningsfulle i faget de brukes i.
- **Læringsutbytte:** Tema på oppgavene må samkjøres med læringsutbytte i faget.
- **Deltakelse:** Studentene må delta aktivt.

Referanser

Aktunc, O. (2013). A teaching methodology for introductory programming courses using Alice. *International Journal of Modern Engineering Research (IJMER)*, 3(1), 350-353.

Anderman, E. M. (1992). Motivation and Cognitive Strategy Use in Reading and Writing. *42nd Annual Meeting of the National Reading Conference* (s. 30). San Antonio, Texas: 42nd Annual Meeting of the National Reading Conference.

Barnes, D. J., Fincher, S., & Thompson, S. (1997). Introductory Problem Solving in Computer Science. I G. Daughton, & P. Magee (Red.), *5th Annual Conference on the Teaching of Computing* (ss. 36-39). Dublin, Ireland: 5th Annual Conference on the Teaching of Computing. Hentet fra https://kar.kent.ac.uk/21468/

Berssanette, J. H., & de Francisco, A. C. (2021). Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education. Research, 20*, 201.

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming. *Journal of the Learning Sciences*, 23(4), 561-599. doi:10.1080/10508406.2014.954750

Bonwell, C. C., & Eison, J. A. (1991). *Active Learning: Creating Excitement in the Classroom*. Washington: ERIC Clearinghouse on Higher Education, The George Washington University.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., & Hadfield, S. M. (2005). RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving. *SIGCSE Bull.*, 37(1), 176–180. doi:10.1145/1047124.1047411

Deek, F. P., Kimmel, H., & McHugh, J. A. (1998). Pedagogical Changes in the Delivery of the First-Course in Computer Science: Problem Solving, Then Programming. *Journal of Engineering Education*, 87(3), 313-320. doi:https://doi.org/10.1002/j.2168-9830.1998.tb00359.x

Edström, K., & Kuttenkeuler, J. (2020, Januar 3). *The teaching trick - How to improve student learning without spending more time teaching*. Hentet fra NTNU: https://www.ntnu.no/documents/1286373847/1305761996/2021-10-25_Teaching-Trick.pdf/24b9b8df-6df2-f0f7-3db2-e6f38d56f76b?t=1638460895011

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences, 111*(23), 8410-8415. doi:10.1073/pnas.1319030111

Kantardjiev, K. O. (2019). *Studentaktiv læring og diversitet – hva fungerer og hvorfor?* Oslo: NOKUT. Hentet fra https://www.nokut.no/globalassets/nokut/rapporter/ua/2019/kantardjiev_studentaktiv-laring-og-diversitet_11-2019.pdf

Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016). The Influence of Problem Solving Abilities on Students' Performance on Different Assessment Tasks in CS1. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (ss. 329–334). Memphis, Tennessee: Association for Computing Machinery. doi:10.1145/2839509.2844596

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016). Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (ss. 1449–1461). San Jose, California: Association for Computing Machinery. doi:10.1145/2858036.2858252

Mathew, R., Iqbal Malik, S., & Tawafak, R. (2019). Teaching Problem Solving Skills using an Educational Game in a Computer Programming Course. *Informatics in Education*, 18, 359-373. doi:10.15388/infedu.2019.17

Nikolic, S., Ros, M., & Hastie, D. B. (2018). Teaching programming in common first year engineering: discipline insights applying a flipped learning problem-solving approach. *Australasian Journal of Engineering Education*, 23(1), 3-14. doi:10.1080/22054952.2018.1507243

Palumbo, D. B. (1990). Programming Language/Problem-Solving Research: A Review of Relevant Issues. *Review of Educational Research, 60*(1), 65-89. doi:10.3102/00346543060001065

Prince, M. (2004). Does Active Learning Work? A Review of the Research. *Journal of Engineering Education, 93*(3), 223-231.

Soloway, E., & Ehrlich, K. (1984). Empirical Studies of Programming Knowledge. *IEEE Transactions on Software Engineering, SE-10*(5), 595-609. doi:10.1109/TSE.1984.5010283

Soloway, E., & Spohrer, J. C. (1988). *Studying the Novice Programmer*. Hillsdale, NJ: L. Erlbaum Associates Inc.

Tellefsen, C. W. (2018). *Studentaktive læringsformer*. Kompetansesenteret for realfagsundervisning, MN, UiO. Oslo, Norge: Universitetet i Oslo. Hentet fra https://www.mn.uio.no/om/organisasjon/adm/prosjekter/interact/aktiv-lering/studentaktive-laeringsformer-notat.pdf

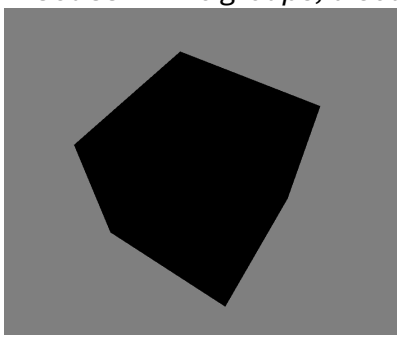
Wieman, C. E. (2014). Large-scale comparison of science teaching methods sends clear message. *Proceedings of the National Academy of Sciences, 111*(23), 8319-8320. doi:10.1073/pnas.1407304111

Vedlegg

Vedlegg 1: Grubleoppgaver

Grubleoppgave: Planlegging
Scenario: *You are asked to create a simulator of a small robot dog, Bittle (see figure below). The simulator should be as close to the real Bittle as possible, both in size and behavior. The simulator will be used to test functionality and limitations in order to predict performance.*
Discuss: *In the groups, discuss what you would do (and should do) before starting to code*

Grubleoppgave: Debugging
Scenario: *You are creating a program that should show a red cube on your screen. The problem is that when you run the program the cube shows up black, like the figure below. You know that you have added light to the scene, and you have set the color to red.*
Discuss: *In the groups, discuss different ways to debug this problem. Be creative!*



Vedlegg 2: Muntlig oblig

Multiple heads one tale

In the specified groups, create a presentation of a given lab that includes:

1. *The theory behind the lab.*
2. *Preliminary thoughts.*
    a. *How did you go about solving the lab?*
    b. *Did you do it differently in the group?*
3. *Implementation.*
    a. *Pseudo code.*
    b. *Diagrams.*
    c. *Did you implement it differently within the group?*
4. *Debugging.*
    a. *Did you have any problems in the coding process?*
    b. *What type of debugging did you do?*

Vedlegg 3: Evalueringsskjema

Spørsmål	**Underspørsmål**	**Svartype**
1. Give your evaluation of the brain teasers:	**1.1** The relevance of the topics (planning and debugging).	Envalgsspørsmål
	1.2 The relevance for the remaining programming courses.	Envalgsspørsmål
	1.3 The learning outcome.	Envalgsspørsmål
	1.4 The format of the activity.	Envalgsspørsmål
	1.5 The length of the activity.	Envalgsspørsmål
	1.6 The explanation of the activity.	Envalgsspørsmål
2. What did you think about the brain teasers in general?		Tekstboks
3. Give your evaluation of "Multiple heads one tale!" (muntlig oblig):	**3.1** The relevance of the task.	Envalgsspørsmål
	3.2 The format of the activity.	Envalgsspørsmål
	3.3 The group sizes.	Envalgsspørsmål
	3.4 The time limit.	Envalgsspørsmål
	3.5 The learning outcome.	Envalgsspørsmål
4. What did you think about "Multiple heads one tale!" in general?		Tekstboks