Norsk informatikkolympiade 2015–2016 — 1. runde

Sponset av

UNIVERSITETET I BERGEN
Institutt for informatikk

FFI Forsvarets forskningsinstitutt

bouvet

Uke 46, 2015

Tid: 90 minutter
Tillatte hjelpemidler: Kun skrivesaker. Det er *ikke* tillatt med kalkulator eller trykte eller håndskrevne hjelpemidler.
Instruksjoner: Oppgavesettet består av 16 oppgaver, med fire svaralternativer på hver oppgave. Det er kun ett riktig svar på hver oppgave. Du får fire poeng for hvert riktige svar, null poeng for feil svar, og ett poeng for hver oppgave du ikke svarer på (det vil si at det ikke lønner seg å gjette dersom du ikke vet hva svaret er). Du kan godt krysse av på alternativene i oppgaveteksten underveis, men du *må* føre inn svarene på svararket helt bakerst.
Oppgavene som handler om programmering starter med beskrivelser av temaet de handler om, slik at de som ikke har vært borti programmering før også kan prøve seg på disse oppgavene. De som kan programmere trenger ikke å lese disse beskrivelsene; all koden oppfører seg som i vanlige programmeringsspråk.

Navn: _____

Skole: _____

Studieretning og årstrinn: _____

Hvor gammel er du 30. juni 2016? _____

Epostadresse: _____

Telefonnummer: _____

1. Du skal fylle 10 bokser med X'er eller O'er. Første boks må inneholde X og siste boks må inneholde O. På hvor mange forskjellige måter kan du fylle boksene?
   Et lovlig eksempel er vist under.

X	O	X	X	X	O	X	O	X	O

   A. 45

   B. 56

   C. 256

   D. 1022

2. En bøtte inneholder 5 røde, 6 grønne, og 7 blå kuler. Hvor mange kuler må du plukke fra bøtta (uten å se) før du er sikker på at du har to kuler med samme farge

   A. 4

   B. 5

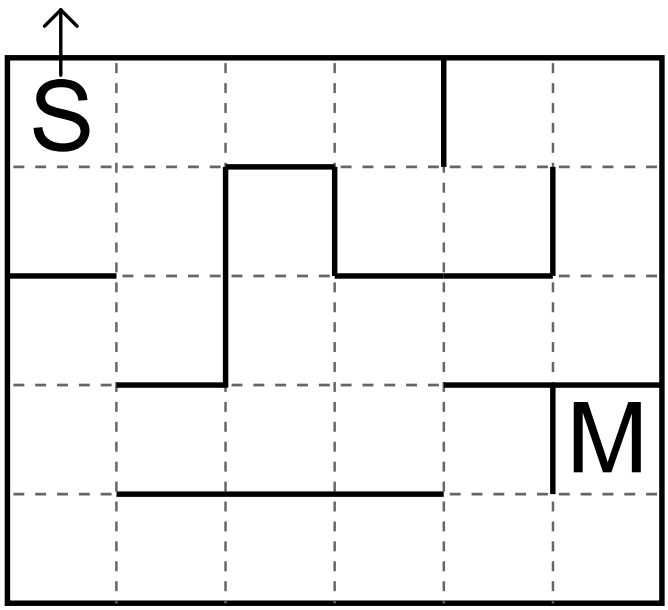
   C. 6

   D. 7

3. En robot kan instrueres til å bevege seg med følgende kommandoer:

   - `F`: flytt en rute framover
   - `V`: sving 90 grader til venstre på stedet
   - `H`: sving 90 grader til høyre på stedet

   Roboten befinner seg i utgangspunktet i ruten merket med S under og ser i retningen pilen peker.



De tykke strekene er vegger som roboten ikke kan bevege seg gjennom. Hva er det minste antall instruksjoner som trengs for at roboten skal komme til ruten markert med M?

A. 16

B. 18

C. 20

D. 22

4. Vi har en maskin som kan brukes til å lage ord. Til å begynne med så inneholder maskinen ordet `A`, og vi kan trykke på følgende knapper for å modifisere ordet:

   - `x` legg til en `A` i begynnelsen av ordet. F.eks. hvis maskinen inneholder ordet `AAB` før operasjonen, så vil den inneholde ordet `AAAB` etter operasjonen.
   - `y` legg til en `B` i slutten av ordet. F.eks. hvis maskinen inneholder ordet `AAB` før operasjonen, så vil den inneholde ordet `AABB` etter operasjonen.
   - `z` speilvend ordet. F.eks. hvis maskinen inneholder ordet `AAB` før operasjonen, så vil den inneholde ordet `BAA` etter operasjonen.

   Hvilket ord vil maskinen ha etter at vi har trykket på knappene i rekkefølge:

   ```
   x x y z y y z y y x z
   ```

   A. $BBBAAABBA$

   B. $BBAAABBBA$

   C. $BBBAABBA$

   D. $BBAABBBA$

5. Gitt maskinen i forrige oppgave, som inneholder utgangsverdien `A`, hvor mange knapper er det minste man må trykke på for å få den til å inneholde `BBAABBAABB`:

   A. 9

   B. 10

   C. 11

   D. 12

6. I det binære tallsystemet bruker man kun to forskjellige sifre: 0 og 1. Sifferet lengst til høyre er 1'er plassen, til venstre for dette har man 2'er-plassen, deretter 4'er-plassen, 8'er-plassen, 16-plassen, og så videre. Hvert siffer er altså dobbelt så mye verdt som sifferet det står til venstre for. Verdien av et tall i det binære tallsystemet finner man ved å summere verdiene for posisjonene som har sifferverdi 1. For eksempel så tilsvarer det binære tallet 10110 verdien $16 + 4 + 2$, altså 22. Hvordan skriver man tallet 220 i binært?

   A. 10111111

   B. 10011010

   C. 11001110

   D. 11011100

7. I *boolsk logikk* jobber vi kun med verdiene **true** og **false**. Vi skal se på fire operatorer som kan brukes til å lage boolske *uttrykk* (formler):

   - NOT-operatoren gir ut motsatt sannhetsverdi av det man mater inn i den: NOT x blir **true** dersom x er **false**, og **false** dersom x er **true**.
   - AND-operatoren brukes med to verdier: x AND y, og gir ut **true** bare hvis både x og y er **true**.
   - OR-operatoren brukes slik: x OR y, og gir ut **true** så lenge minst én av x og y (eller begge to) er true. Altså gir den ut **false** bare hvis både x og y er **false**.
   - XOR-operatoren brukes slik: x XOR y, og gir ut **true** så lenge én av x og y er **true** og den andre er **false**. Hvis x og y har samme verdi, gir XOR ut **false**.

   Disse operatorene kan kombineres til større uttrykk. Parenteser angir rekkefølgen uttrykket skal *evalueres* (regnes ut) i, slik som vi kjenner fra matematikken.

   I hvilke tilfeller er uttrykket

   $$(x \text{ OR } y) \text{ XOR } ((\text{NOT } x) \text{ OR } y)$$

   sant?

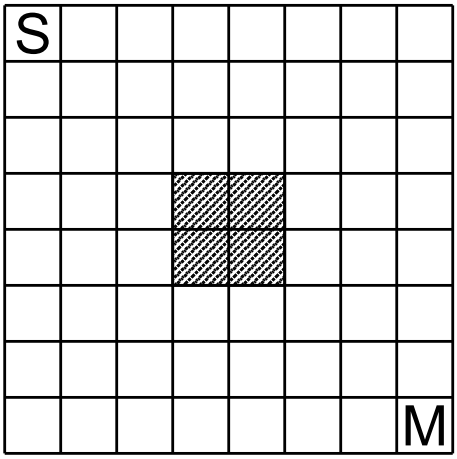
   A. alltid når $x =$ **true**

   B. alltid når $y =$ **false**

   C. bare når $x =$ **true** og $y =$ **false**

   D. bare når $x =$ **false** og $y =$ **false**

8. Tegningen under viser et kvadratisk torg bestående av 60 steinheller med en liten gressslette i midten (det skraverte området).



Du ønsker å gå fra hellen som er øverst til venste til hellen som er nederst til høyre (markert med henholdsvis S og M). Du har kun lov til å flytte deg én helle nedover eller én helle til høyre for hvert skritt. Du har heller ikke lov til å tråkke på gressletten. Hvor mange forskjellige stier kan du ta?

A. 434

B. 492

C. 876

D. 982

9. I de fleste programmeringsspråk brukes operatoren `=` på en annen måte enn i matematikken. Den instruerer nemlig datamaskinen om å regne ut verdien av uttrykket på høyre side av likhetstegnet og legge den inn i variabelen på venstre side. (Verdien *kopieres* alltid, den *flyttes* ikke.) En variabel holder på en verdi helt frem til du legger noe annet inn i den; da forsvinner den gamle verdien. Linjer som står etter hverandre utføres etter tur. For eksempel vil

```
x = 5
x = x + 3
```

gjøre at variabelen x ender opp med å inneholde verdien 8. Hva blir verdien av x etter at følgende kode er utført?

```
x = 9
x = 2 + (3 * x)
x = 50 - x
```

A. 9

B. 21

C. 27

D. 29

10. En *funksjon* er en navngitt samling med programinstruksjoner, som kan *kalles* (startes) med *inndata* og *returnere* (gi tilbake) *utdata*. Kommandoen `if` sjekker om uttrykket som står i parenteser er **true** (sann) eller **false**(usann); hvis det er **true**, gjøres det som står i krøllparentesene etter `if`; hvis det er **false**, gjøres det som står i krøllparentesene etter den tilhørende `else`'n. `return` avslutter funksjonen og returnerer resultatet av uttrykket som står på samme linje. `<` er den vanlige mindre-enn-operatoren — f.eks. vil `3 < 4` bli **true**, og `4 < 3` blir **false**. `3 < 3` blir også **false**. Hva returnerer funksjonen under dersom den kalles `f(9, 11)`? (altså slik at `A` har verdien 9 og `B` har verdien 11)

```
f(A, B) {
        if (A < B) {
                B = B + A
                A = B
        } else {
                A = 2 * A
        }

        if (B < A) {
                A = A + 5
        } else {
                A = A + B
        }

        return A + B
}
```

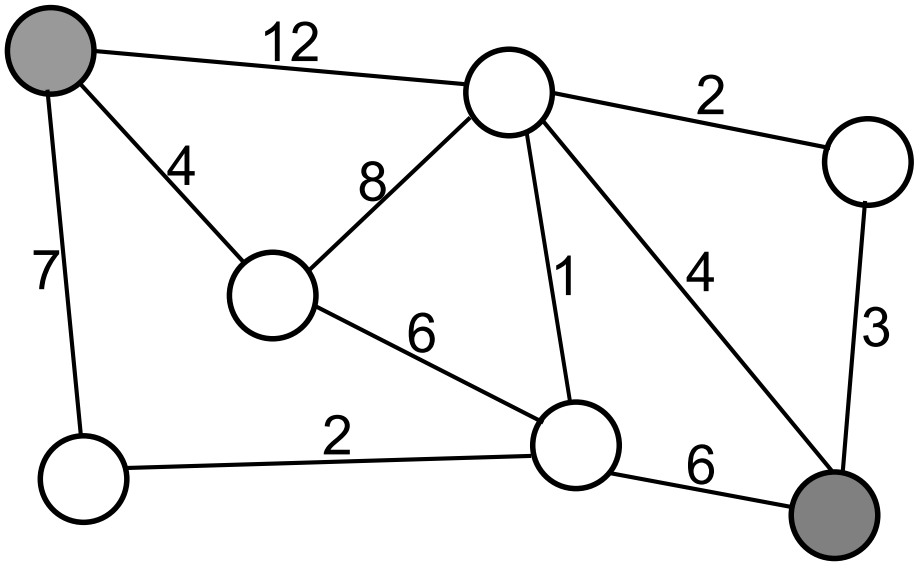
A. 20

B. 44

C. 51

D. 60

11. I datasammenheng er en *graf* en samling av *noder* (punkter; her tegnet som sirkler) og *kanter* (streker) mellom nodene. Vi vil i denne oppgaven se på grafer der hver kant har en *lengde*, som her er markert med tallene ved siden av strekene (lengden av kanten har ingen sammenheng med den "fysiske" lengen av streken den er tegnet med). *Avstanden* mellom to noder er den totale lengden av den korteste stien (sekvensen av kanter) mellom dem.



Hva er avstanden mellom de to fargelagte nodene i tegningen over?

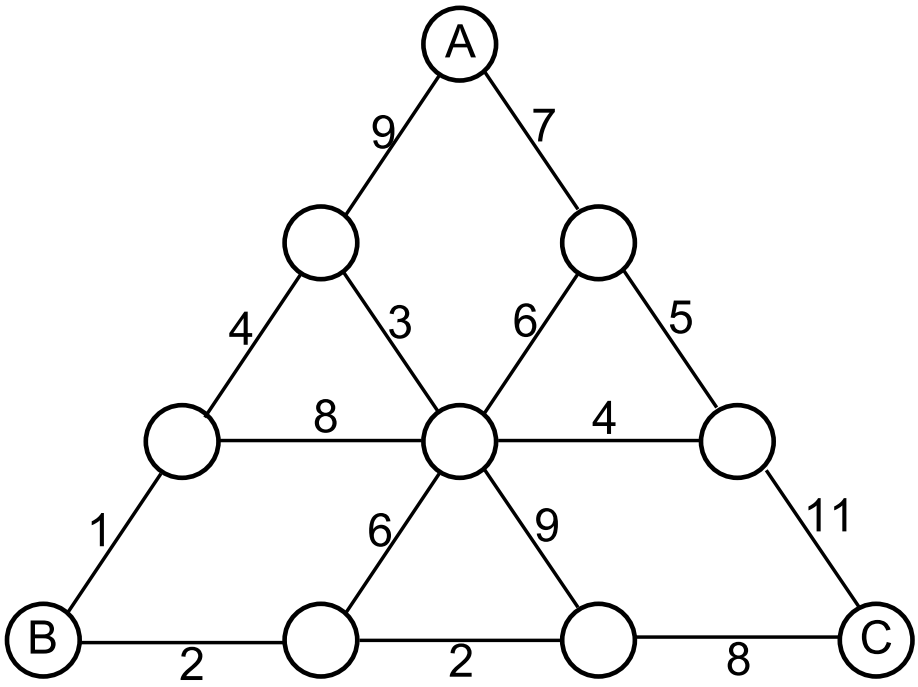
A. 13

B. 14

C. 15

D. 16

12. Anne, Bente og Cecilie befinner seg henholdsvis i nodene markert med A, B og C i grafen under. De ønsker å treffes i en av nodene i grafen. (Denne noden kan være en hvilken som helst - den trenger ikke være en av A, B eller C.) Dersom lengen på hver kant er antall minutter det tar å reise mellom nodene den forbinder, hva er minste mulig tid som gjør at alle jentene kan samles på samme node?



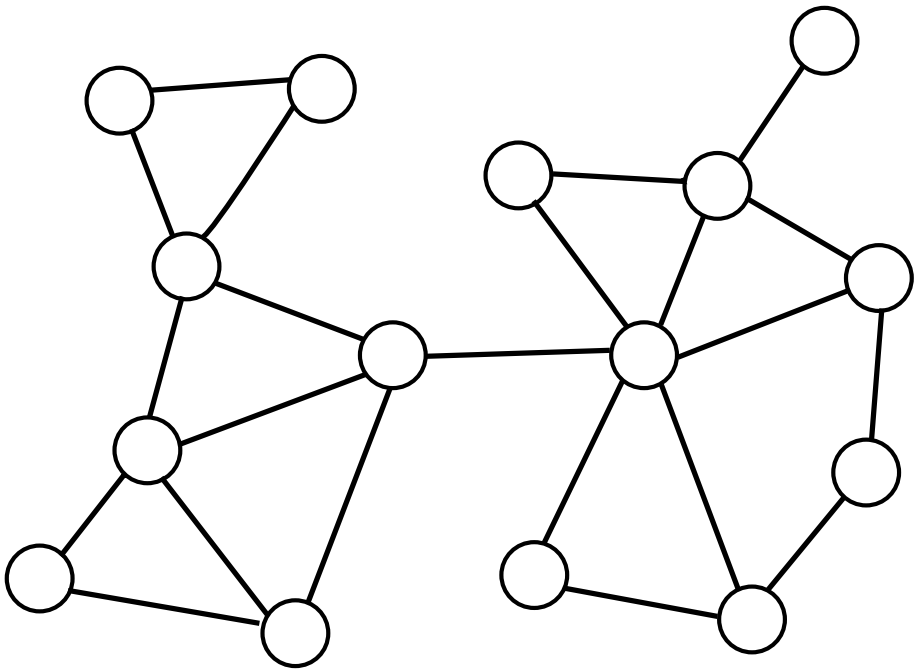
A. 10 minutter

B. 11 minutter

C. 12 minutter

D. 15 minutter

13. Grafen under viser vennskap på det sosiale nettverket NIOBok. Nodene representerer medlemmer, og kantene viser hvilke medlemmer som er venner.



Dersom noen poster (legger ut) en lenke til en nettside så vil den være tilgjengelig for personen som legger ut lenken og for alle medlemmene som er direkte venner med den personen på nettverket. (Folk som bare er venner av venner av den som legger ut lenken vil ikke kunne se den.) Du ønsker å få flere besøkende til `http://nio.no/pensum-til-andre-runde-i-nio/` og prøver derfor å få folk på NIOBok til å poste denne lenken. Hva er det minste antall personer du må få til å poste lenken for at alle medlemmene på nettverket skal få tilgang til den?

A. 3

B. 4

C. 5

D. 6

14. **Merk:** De siste tre oppgavene vil være ekstra utfordrende for dem som ikke har erfaring med programmering fra før.

Funksjoner i programmeringsspråk kan også inneholde kall til seg selv. Dette kalles *rekursive* funksjoner. Når en rekursiv funksjon kaller seg selv, starter en ny utgave av den samme funksjonen, og den gamle utgaven venter til den nye er blitt ferdig. Gitt følgende definisjon av funksjonen `f`:

```
f(x) {
    if (x < 2) {
        return x
    } else {
        return 2 * f(x - 1) + f(x - 2)
    }
}
```

Hva returnerer kallet `f(7)`?

A. 13

B. 43

C. 169

D. 328

15. En *while-løkke* utfører koden inni krøllparentesene sine gjentatte ganger. Før hver utførelse sjekker løkken betingelsen inni parentesene rett etter `while`. Dersom betingelsen er usann, avsluttes løkken, og man fortsetter med koden etter avslutningskrøllparentesen (dersom det er noe kode der i det hele tatt).

    Et *array* er en samling med et bestemt antall elementer. `A[0]` er det første elementet, `A[1]` er det andre osv. Antallet elementer totalt er `length(A)`. Det siste elementet er dermed `A[length(A) - 1]`.

```
f(A) {
    best = 0
    i = 0
    while (i < length(A)) {
        sum = 0
        j = i
        while (j < length(A)) {
            sum = sum + A[j]
            if (sum > best) {
                best = sum
            }
            j = j + 1
        }
        i = i + 1
    }
    return best
}
```

Dersom funksjonen kalles med en array `A` hvor `length(A)` = 100, hvor ofte vil linjen markert med firkant bli utført?

A. 4950

B. 4975

C. 5000

D. 5050

16. Funksjonene `f` og `g` er definert som følger

```
f(x) {
    s = 0
    a = 1
    if (x > 0) {
        s = f(x-1)
    } else {
        s = 1
    }

    while (x > 0) {
        a = 2 * a
        x = x - 1
    }
    return s + g(a, 2*a)
}
g(x, y) {
    s = 0
    while (x < y) {
        s = s + 2*(y - x) - 1
        x = x + 1
    }
    return s
}
```

Hvis `t = f(10)`, hva er siste siffer i `t`?

A. 0

B. 1

C. 2

D. 6

Svarark

Oppgave	**A**	**B**	**C**	**D**	Poeng (til bruk for læreren)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
Poengsum					

Til læreren: Husk at korrekt svar gir 4 poeng, feil svar gir 0 poeng, og fraværende svar gir 1 poeng.