Norsk informatikkolympiade 2017–2018 — 1. runde

Sponset av

UNIVERSITETET I BERGEN
Institutt for informatikk

Uke 46, 2017

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 2018? _____

Epostadresse **(skriv tydelig)**: _____

Telefonnummer: _____

1. Du har 12 byer som du ønsker å forbinde med hurtigtoglinjer på en slik måte at det vil være mulig å reise mellom alle par av byer ved å ta ett eller flere hurtigtog. Hvor mange toglinjer må du da konstruere? En toglinje går bare fra en by til en annen (og tilbake igjen) uten å stoppe i andre byer underveis, men man kan bytte tog i en by dersom den har flere toglinjer.

   A. 6

   B. 11

   C. 12

   D. 66

2. Du har 1000 kronestykker. En av myntene er falsk og veier litt mindre enn de andre myntene. Du har en vekt hvor du kan plassere så mange mynter du vil på hver av to vektskåler, og den vil fortelle deg hvilken av haugene som er lettest - eller om de veier like mye. Hvor mange veiinger trenger du for å finne ut hvilken mynt som er falsk?

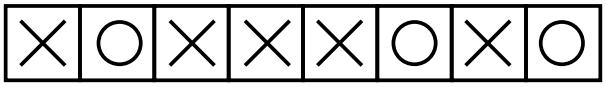
   A. 6

   B. 7

   C. 10

   D. 500

3. Du har 8 bokser som står på rekke. I hver boks skal du skrive enten `X` eller `O`. På hvor mange måter kan du fylle inn boksene slik at det ikke er to `O`-er i nabobokser? Ett eksempel er vist under.



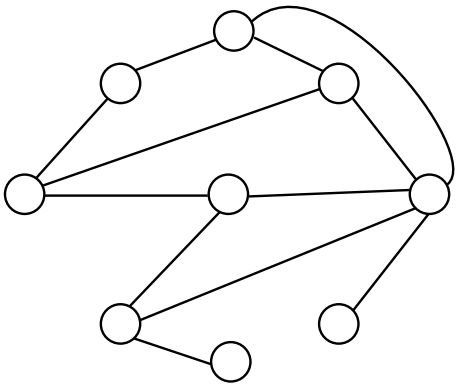
   A. 55

   B. 63

   C. 89

   D. 128

4. I datasammenheng er en *graf* en samling av *noder* (punkter; her tegnet som sirkler) og *kanter* (streker) som forbinder nodene. En node er en *nabo* av en annen node dersom de er forbundet med en kant. Hvor mange av nodene i grafen under har nøyaktig 3 naboer?



A. 3

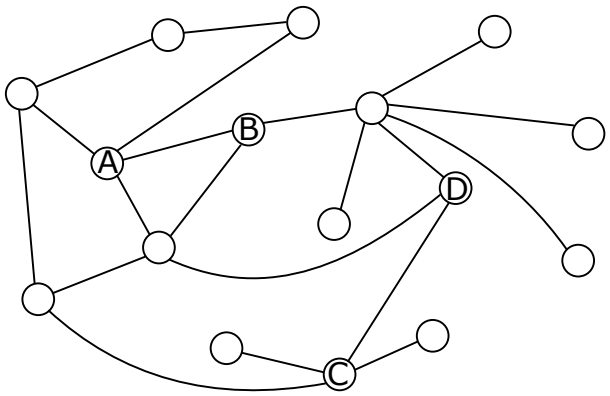
B. 4

C. 5

D. 6

5. NIOBok er et populært sosialt nettverk hvor medlemmene kan dele informasjon om algoritmer og datastrukturer med vennene sine. Når et medlem legger ut et innlegg så kan alle som er venner med dette medlemmet se innlegget, og både *like* og *dele* innlegget. Dersom noen *deler* et innlegg så kan også vennene til den personen like innlegget, men de kan ikke dele videre. Det er også også mulig å like sitt eget innlegg. Man ikke like det sammee innlegget flere ganger, selv om flere man kjenner har delt det.

   Nodene i grafen under representerer medlemmer på NIOBok, og kantene representerer hvem som er venner. Dersom alle medlemmene deler og liker alle innleggene de kan, hvem av A, B, C og D vil få flest "liker" på innleggene sine?



   A. A

   B. B

   C. C

   D. D

6. I det binære tallsystemet så bruker man kun to 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 11010 verdien $16 + 8 + 2$, altså 26. Hvilket av de binære tallene under er et primtall? (Et primtall er et heltall som bare er delbart med seg selv og 1, uten at man får desimaler i resultatet.)

   A. 100111

   B. 101101

   C. 111010

   D. 111101

7. En *bitsekvens* er en følge med 0'er og 1 ere. F.eks. så er 01100 en bitsekvens med lengde 5. Hvor mange bitsekvenser finnes det med lengde 10 som inneholder nøyaktig 3 1-ere?

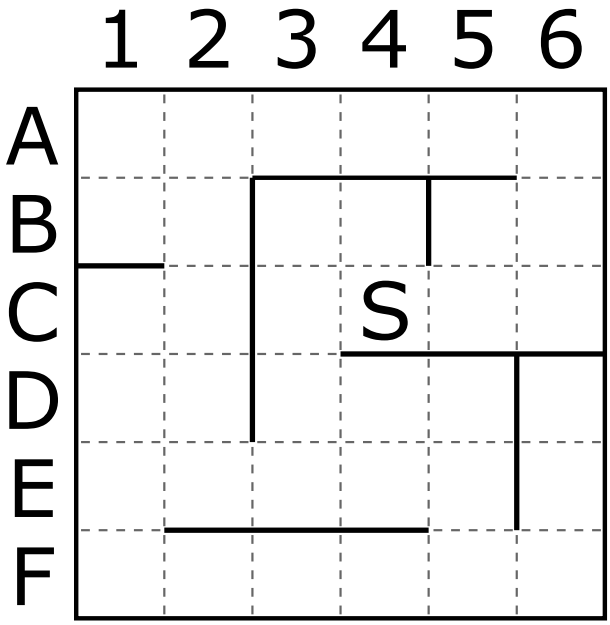
   A. 120

   B. 512

   C. 720

   D. 1024

8. Robot-designerene på Universitetet i Begen har konstruert en robot som beveger seg basert på følgende tre kommandoer: `F`: fram en rute; `V`: sving 90 grader til venstre på stedet; `H`: sving 90 grader til høyre på stedet. Et *program* består av en kjede med kommandoer som roboten skal utføre i rekkefølge. Dersom en `F` kommando vil få roboten til å kjøre inn i en vegg så vil roboten ignorere denne kommandoen og fortsette på neste. Roboten befinner seg til å begynne med i rute C4 på kartet (merket med bokstaven S) med fronten rettet nordover (oppover på arket):



   På hvilken rute vil roboten stå etter å ha utført følgende program?

   ```
   F V F F V F F F H F F F V F H H F F H
   ```

   A. C2

   B. F1

   C. D1

   D. A3

9. For å kunne lagre programmer for denne roboten på en effektiv måte så har robot-designerene kommet opp med følgende måte å oversette mellom instruksjoner og bitsekvenser, slik at hver instruksjon er på nøyaktig to bits.

Bitsekvens	Instruksjon
`00`	`V`
`01`	`H`
`10`	`F`

Du ser her at sekvensen `11` ikke blir brukt til noe. Veldig ofte så vil et program for roboten ha behov for mange `F`-operasjoner etter hverandre. Designerene har derfor besluttet at denne sekvensen skal brukes til å representere flere `F`-operasjoner, avhengig av hva de neste to bitsene er. Det endelige settet med instruksjoner ble dermed

Bitsekvens	Instruksjoner
`00`	`V`
`01`	`H`
`10`	`F`
`1100`	`FF`
`1101`	`FFF`
`1110`	`FFFF`
`1111`	`FFFFF`

F.eks. vil da `F H FFFF FFF` kunne lagres som `10 01 1110 1101`.

Du ønsker å mest mulig effektivt lagre programmet som sier `F` 44 ganger, etterfulgt av én `H`, etterfulgt av `F` 31 ganger, etterfulgt av én `V`, etterfulgt av `F` 20 ganger. Hvor mange bits trenger du for det?

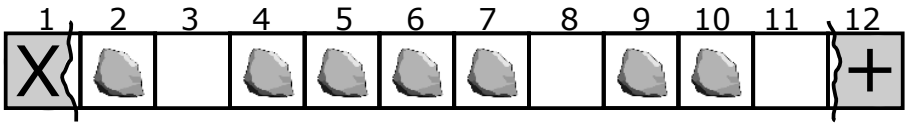
A. 78

B. 80

C. 82

D. 84

10. Tegningen under viser en elv hvor det befinner seg noen steiner. Du befinner deg på elvebredden på venstre side (rute 1, merket med `X`) og ønsker å krysse over til høyre side (rute 12, merket med `+`) ved å gjøre en rekke hopp. Du kan kun hoppe til høyre og kan ikke hoppe lengre enn 3 ruter av gangen. Du kan heller ikke lande på ruter i elven hvor det ikke er noen stein. Dvs. at fra start-posisjonen kan du kun hoppe til rute 2 og 4. På hvor mange forskjellige måter kan du hoppe for å krysse elven?



A. 40

B. 44

C. 46

D. 50

11. 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, selv hvis høyresiden er en enkelt variabel — 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 = 4
x = x + 3
```

gjøre at variabelen `x` ender opp med å inneholde verdien 7.

Hva blir verdien av `x` etter at følgende kode er utført?

```
x = 6
x = 8 - x
x = 3 * x
```

A. 2

B. 6

C. 8

D. 18

12. En *funksjon* er en navngitt samling med programinstruksjoner, som kan *kalles* (startes) med *inndata* og *returnere* (gi tilbake) *utdata*. Hva returnerer funksjonen under dersom den kalles som `f(8, 4)`? Merk at den første linjen bare oppgir navnet på funksjonen (`f`) og navnene på inndataene (`a` og `b`).

```
f(a, b) {
    c = a + b
    a = c + b
    b = c + a
    return a + b
}
```

A. 12

B. 36

C. 44

D. 52

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

    Kommandoen `if` sjekker om uttrykket som står i parenteser er **true** eller **false**; 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 som `f(19, 8, 7, 12)`?

```
f(a, b, c, d) {
    if (a < c) {
        c = a
    } else {
        a = c
    }

    if (b < d) {
        d = b
    } else {
```

```
        b = d
    }

    if (a < b) {
        b = a
    } else {
        a = b
    }

    return a
}
```

A. 19

B. 8

C. 7

D. 46

14. 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,y) {
    if (x == 0) {
        if (y < 0) {
            return 0
        } else {
            return y
        }
    } else {
        return f(x-1, y) + f(x-1, y-1) + f(x-1, y-2)
    }
}
```

Merk at `==` sjekker om to verdier er like.

Hva returnerer funksjonskallet `f(3,7)`?

A. 36

B. 45

C. 81

D. 108

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]`.

    Funksjonen `sorter` under sorterer en array med heltall ved å gjøre kall til `bytt`, og returnerer antall kall til `bytt` som ble gjort.

```
bytt(A, p, q) {
    x = A[p]
    A[p] = A[q]
    A[q] = x
}

sorter(A) {
    byttinger = 0
    i = 0
    while (i < length(A)) {
        if (i > 0) {
            if (A[i] < A[i-1]) {
                bytt(A, i, i-1)
                byttinger = byttinger + 1
                i = i - 1
            } else {
                i = i + 1
            }
        } else {
            i = i + 1
        }
    }
    return byttinger
}
```

    Hvis `length(A)`$= 100$, hva er da største mulig verdi som `sorter(A)` kan returnere.

    A. 99

    B. 4950

    C. 9801

    D. 9900

16. Gitt følgende definisjon av funksjonene `d`, `u` og `e`:

```
d(x, a) {
    l = 0
    while(l < x) {
        l = l + a
    }
    if (l == x) {
        return a
    } else {
        return 0
    }
}

u(x) {
    m = 0
    s = 0
    while (m < x) {
        m = m + 1
        s = s + d(x, m)
    }
    return s
}

e(n) {
    x = 1
    s = 0
    while (n > 0) {
        x = x * 3
        n = n - 1
        s = s + u(x)
    }
    return s
}
```

Hva returnerer kallet `e(5)`?

A. 542

B. 1024

C. 1635

D. 2017

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.