INFT2503 Øving 02

Oppgave 2

> Hva vil skje hvis du skriver:
>
> ```
> char *line = nullptr;   // eller char *line = 0;
> strcpy(line, "Dette er en tekst");
> ```

Her defineres en peker `*line` til å peke til en nulladresse. Det er ikke allokert noe minne til denne, siden den er tom og peker til `null`, altså ingen steder.

Når vi da forsøker å kopiere strengen `"Dette er en tekst"` inn på minneplasseringen til `line`(som er null) vil det si av vi forsøker å kopiere en streng inn på et minneområde vi ikke ahr tilgang til, siden det ikke er allokert noe minne til `line`. Da vil vi få en feilmelding av typen `segmentation fault` eller `SIGSEGV`.

Segmentation fault får vi som regel når vi prøver å lese eller skrive til en ugyldig minneadresse, peke til en ugyldig minneplassering (f.eks `nullptr`) eller prøver å bruke en peker som ikke er initialisert.

Oppgave 3

> Finn ting som kan gå galt i følgende programbit:
>
> ```
> char text[5];
> char *pointer = text;
> char search_for = 'e';
> cin >> text;
> while (*pointer != search_for) {
>   *pointer = search_for;
>   pointer++;
> }
> ```

`char text[5]` allokerer bare plass til 4 tegn pluss nullterminator `0\`. Altså om bruker skriver inn mer enn 4 tegn kan det føre til buffer overflow, siden input vil forsøke å skrive utenfor grensen til char-arrayet.

I `char *pointer = text;` peker `pointer` på den første adressen adressen i char-arrayet `text`. Denne kan brukes til å gå utenfor grensene til arrayet, og dermed skrive eller lese fra minneområder utenfor det.

Det settes opp en `while`-loop som kjører fra første adresse satt ved `pointer` og helt til den finner en `e`, som definert i `search_for = e`. Hvis den ikke finner karakteren `e` vil den fortsette å inkrementere pekerens adresse "i det udendelige" og dermed går ut over grensen til `text`-arrayet, hvilket fører til at programmet til slutt kræsjer:

```
*** stack smashing detected ***: terminated
Process finished with exit code 134 (interrupted by signal 6:SIGABRT)
```

Oppgave 4

Finn alle syntaksfeil i følgende programbit:

```
int a = 5;
int &b;
int *c;
c = &b;
*a = *b + *c;
&b = 2;
```

Legg setningene inn i en .cpp-fil og endre koden slik at den kompilerer. Beskriv årsaken til hver enkelt av kompileringsfeilene i kommentarer.

Løsning:

```
# include <iostream>
using namespace std;

int main()
{
   int a = 5; // Riktig. Nothing to see here.
   cout << "a = " << a << endl;

   /**
    * int &b;
    * Feil! &b betyr "adressen til b", eller "b er en referanse til [noe]".
    * En referanse må initiliseres når den blir deklarert.
    * Løsning: b må ha en referanse til en eksisterende variabel, f.eks a.
    */
   int b = a;
   cout << "b = a = " << b << endl;
   cout << endl;


   /**
    * int *c;
    * Ingen problem. c peker til int.
    */
   int *c;
   cout << "*c = " << *c << endl;
   cout << endl;


   /**
    * c = &b;
    * Feil! b er en referanse, og vi kan ikke hente ut adressen til en referanse slik.
```

```
     */
    cout << "c = " << c << endl;
    c = &a;
    cout << "c = &a = " << &a << endl;
    cout << endl;


    /**
     * *a = *b + *c;
     * Feil! a er en int, ikke en peker. Den kan ikke bli deferert. b, som nå
     * peker på a er også dermed en int, så samme gjelder for b.
     * Løsning: Fjerne pekernotasjon * foran a.
     */
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "*c = " << *c << endl;
    a = b + *c;
    cout << "a = b + *c" << endl;
    cout << "a = " << b << " + " << *c << " = " << a << endl;
    cout << endl;
    /**
     * Her vil vi få noe rart pga rekkefølgen på utskrift og utregning!
     * Utskriften viser a = 5 + 10 = 10
     * a blir her satt til summen av b og *c, dvs b=5 + c=5.
     * c peker på a, som etter tilordningen = 10. Dermed vil utskriften
     * vise 5 + 10 = 10, fordi den skriver ut verdiene etter at de er oppdatert.
     * Skriver vi ut verdiene FØR de oppdateres ser vi at *c = 5.
     */


    /**
     * &b = 2;
     * Feil! Vi kan ikke tilordne en verdi direkte til en referanse.
     * Løsning: Sett b = 2. Dette vil endre verdien til a, fordi vi
     * satte b til å være en referanse til a.
     * b => a = 2
     */
    b = 2;
    cout << "b = " << b << endl;
}
```