TÖL309G Tölvutækni og forritun

Heimadæmi 2

Hjörvar Sigurðsson

1.

```
localhost:~$ gcc -g -o hallo_heimur hallo_heimur.c
localhost:~$ gbd ./hallo_heimur
/bin/sh: gbd: not found
localhost:~$ gdb ./hallo_heimur
GNU gdb (GDB) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "i586-alpine-linux-musl".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./hallo_heimur...
(gdb) r
Starting program: /home/hjs33/hallo_heimur
Halló heimur[Inferior 1 (process 143) exited normally]
(gdb)
```

2. a) Forritið hrynur þegar i = 8.

Þegar double x er sett á undan a fylkinu, þá þarf ekki að „labba í gegnum“ x breytuna við keyrslu fallsins, þar sem það „sleppir“ x og fer beint í a[0], a[1], d3...d0, d7...d4, o.s.frv..

b) Forritið hrynur þegar i = 10.

Þegar double x er sett fyrir aftan d, þá þarf að „labba í gegnum“ x breytuna líka við keyrslu fallsins, þar sem það byrjar í a[0], a[1], o.s.frv., og fer svo í gegnum d og svo x.

c) Forritið hrynur þegar i = 14.

Við keyrslu fallsins þá er farið í gegnum öll átta stök fylkis a, og svo d. Sjá töflu 2.c. Þegar a er átta staka int fylki, þá eru mikilvæg gögn í minnishólfi 14, en því hrinur forritið þegar i = 14.

Minni	**i**
Mikilvæg gögn	14
(Ónotað)	13
(Ónotað)	12
(Ónotað)	11
(Ónotað)	10
d7 ... d4	9
d3 ... d0	8
a[7]	7
a[6]	6
a[5]	5
a[4]	4
a[3]	3
a[2]	2
a[1]	1
a[0]	0

Tafla 2.c.

3. a) i = 5, og p er bendill sem bendir á minnishólf i.

   b)

   Í (int *p = q) er p bendill á minnishólf q. Gagnatag q þarf að vera int.

   Í (*p = q) er gildið í minnishólfinu sem p bendir á uppfært sem q, en q er gildi af sama tagi og p; t.d. ef p er int *p þá er q int, ef p er double *p þá er q double, o.s.frv..

   c) Það vantar að tilgreina á hvaða minnishólf bendillin p bendir. Rétt væri að segja t.d.:

   ```
   int i = 3;
   int *p = &i;    - q er þá bendill á vistfang breytunnar i
   *p = 2;         - i verður 2
   ```

4. Kóði:

```c
#include <stdio.h>
#include <stdlib.h>

int isprime(long k) {

    if (k == 0 || k == 1) {
        return 0;
    }
```

```
    for (int i = 2; i <= k / 2; i++) {
        if (k % i == 0) {
            return 0;
        }
    }

    return 1;
}

int main(int argc, char *argv[]) {

    if (argc<1) {
        exit(1);
    }
    int n = atoi(argv[1]);


    int fjoldiTwinPrime = 0;

    long current = 0;

    while (fjoldiTwinPrime < n) {
        if (isprime(current) == 1) {
            if (isprime(current + 2) == 1) {
                fjoldiTwinPrime += 1;
                printf("%ld og %ld\n", current, current + 2);
            }
        }
        current += 1;
    }

    return 0;
}
```

Output:

```
localhost:~$ ./twinpr 6
3 og 5
5 og 7
11 og 13
17 og 19
29 og 31
41 og 43
```

5. Kóði:

```c
/*************************************************************
  Beinagrind að lausn á dæmi 5 í heimadæmum 2 í Tölvutækni
  og forritun, haust 2022

  Sjá lýsingu verkefnis á dæmablaði
*************************************************************/
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node *next;
};


/* Prentar út stökin í tengdum list */
void printList(struct Node *h) {
    printf("Listi: ");
    while (h != NULL) {
        printf("%d", h->data);
        h = h->next;
        if (h != NULL) printf(" -> ");
    }
    printf("\n");
}


/* Eyðir hnúti númer k í tengda listanum sem head bendir á.
   Skilar bendi á fremsta hnút listans                    */
struct Node* delNode(struct Node* head, int k) {

    /* Hér þarf að fylla inn með kóða */

    struct Node* tempNodeCurr = (struct Node *)malloc(sizeof(struct Node));
    tempNodeCurr = head;
    struct Node* tempNodeDel = (struct Node *)malloc(sizeof(struct Node));

    // Iterate-a að hnút á undan þeim sem skal eyða.
    for (int i = 0; i < (k - 2); i++) {
        tempNodeCurr = tempNodeCurr->next;
    }

    // Endurtengi listann til að eyða hnút.

    if (k == 1) {
        return head->next;
    }
```

```
    tempNodeDel = tempNodeCurr->next;
    printf("debug: curr: %d\n", tempNodeCurr->data); //debug
    printf("debug: del: %d\n", tempNodeDel->data); //debug
    tempNodeCurr->next = tempNodeDel->next;

    return head;
}


/* Býr til n-staka lista með slembigildum (0 til 99),
   skilar bendi á fremsta hnút, eða NULL ef n < 1 */
struct Node* createList(int n) {
    struct Node *head, *p;
    int i;

    /* Ef n er núll eða minna þá tómur listi */
    if (n < 1) return NULL;

    /* Búa til fyrsta hnútinn og láta head benda á hann */
    head = p = (struct Node *)malloc(sizeof(struct Node));
    head->data = rand()%100;
    head->next = NULL;

    /* Búa til restina af hnútunum */
    for(i=1; i<n; i++) {
        p = (struct Node *)malloc(sizeof(struct Node));
        p->data = rand()%100;
        p->next = head;
        head = p;
    }

    return head;
}


int main(int argc, char **argv) {
    struct Node *list;

    /* Búa til listann með 10 slembigildum */
    list = createList(10);
    printList(list);

    /* Eyða út nokkrum hnútum og skoða listann í hvert sinn */
    printf("debug: Á að eyða hnút 1.\n");
    list = delNode(list, 1);
    printList(list);

    printf("debug: Á að eyða hnút 3.\n");
```

```
    list = delNode(list, 3);
    printList(list);

    printf("debug: Á að eyða hnút 20.\n");
    list = delNode(list, 20);
    printList(list);

    return 0;
}
```

Output (bætti við debug skipunum til þess að auðvelda fyrir mig):

```
localhost:~$ ./delete
Listi: 52 -> 28 -> 0 -> 54 -> 50 -> 88 -> 55 -> 95 -> 66 -> 0
debug: Á að eyða hnút 1.
Listi: 28 -> 0 -> 54 -> 50 -> 88 -> 55 -> 95 -> 66 -> 0
debug: Á að eyða hnút 3.
debug: curr: 0
debug: del: 54
Listi: 28 -> 0 -> 50 -> 88 -> 55 -> 95 -> 66 -> 0
debug: Á að eyða hnút 20.
Segmentation fault
localhost:~$
```