

سوال سوم تمرین پنجم مبانی برنامه سازی:

کد اول:

```
C 3.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main()
4  {
5      int *container;
6      container = (int *)malloc(40);
7      printf("%d\n", sizeof(container));
8      free(container);
9      return 0;
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\آموزش\1 تریم دانشگاه> gcc 3.c
PS D:\آموزش\1 تریم دانشگاه> ./a

4

این کد با ایجاد پوینتری از جنس int^* ۴۰ بایت از حافظه را به آن الوکیت می کند اما نکته ای که هست این است که در هنگام چاپ کردن خروجی برنامه سائز خود پوینتر container (یعنی سائز خود متغیر پوینتر که آدرس در آن نگهداری می شود و نه مقدار حافظه ای که به آن الوکیت شده!) را چاپ می کند که به دلیل ۳۲ بیتی بودن سیستم کامپیوتر من (قدیمی!) خروجی مقدار ۴ بایت را نشان می دهد.

کد دوم:

```
C 3.c > main()
1  #include <stdio.h>
2  int main()
3  {
4      int i = 22265484213,*pointer;
5      pointer= &i;
6      void *void_ptr;
7      void_ptr = &pointer;
8      printf("\nValue of iptr = %d ", **(int *)void_ptr);
9      return 0;
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\ترم 1\آموزش برنامه سازی> gcc 3.c

3.c: In function 'main':

3.c:4:14: warning: overflow in conversion from 'long long int' to 'int' changes value from '22265484213' to '790647733' [-Woverflow]

```
4 |     int i = 22265484213,*pointer;
  |             ^~~~~~
```

PS D:\ترم 1\آموزش برنامه سازی> ./a

Value of iptr = 790647733

اولین موضوع راجب کد طبق warning این است که مقدار ذخیره شده در متغیر i که از جنس int است از مقدار مجاز آن بیشتر است و مقدار آن اورفلو می کند. در این کد اول پوینتری از جنس `int*` تعریف می شود که به متغیر i اشاره می کند و بعد از آن پوینتر دیگری از جنس `void*` ایجاد می شود که به پوینتر قبلی که حاوی آدرس i است اشاره می کند. پوینتر های از جنس `void*` نشان دهنده نوع خاصی از داده نیستند و صرفاً به خانه ای از حافظه اشاره می کنند و به همین ترتیب قابل dereference کردن به تنهایی نیستند و قبل از آن باید به نوعی دیگر از پوینتر کست شوند. در این کد با کست شدن `void_ptr` به `int` مقدار زیادی ستاره! (تنها مهم است که تعداد ستاره ها بیشتر یا برابر دو باشند) و بعد از آن دوبار dereference کردن آن (در اولین دفعه به آدرس i و در دومین دفعه به مقدار درون آدرس i) به همان مقدار تغییر یافته متغیر i می رسیم که در خروجی چاپ می شود.

کد سوم:

```
C 3.c > main()
1 #include <stdio.h>
2 int main()
3 {
4     char *a = "kevin";
5     printf("%carl?"
6           "%s Dave!" + 1
7           ,a
8           , "bob\0\b stuart");
9 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\آموزش\ترم\1\ترم\آموزش\3.c gcc 3.c
PS D:\آموزش\ترم\1\ترم\آموزش\3.c ./a
carl?kevin Dave!

در این کد ابتدا پوینتری از جنس `char*` تعریف می شود که به خانه ی اول استرینگ `kevin` از حافظه اشاره می کند.

به سراغ فرمت استرینگ پیچیده ی درون کد می رویم! با جمع کردن فرمت استرینگ با یک عملا به اندازه ی یک بایت (به اندازه ی یک کاراکتر) آن را به جلو شیفت داده ایم که یعنی کاراکتر % از ابتدای آن حذف می شود و آن را از کاراکتر دوم می خوانیم که به صورت زیر خواهد بود:

“carl?””%s Dave!”

حال رشته ی `a` به جای `%s` قرار می گیرد و آرگومان بعدی `printf` که رشته ی دیگری است اصلا چاپ نمی شود که خروجی به صورت زیر خواهد بود:

carl?kevin Dave!

نکته ای که هست این است که به نظر می رسد در شرایطی شبیه زیر برنامه فقط رشته های اول را چاپ می کند و اخطار می دهد که آرگومان های تابع بیش از اندازه هستند.

```
C 3.c > main()
1 #include <stdio.h>
2 int main()
3 {
4     printf("Ar"vin", "Baghal Asl");
5 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\آموزش\ترم\1\ترم\آموزش\3.c gcc 3.c
PS D:\آموزش\ترم\1\ترم\آموزش\3.c ./a
Arvin