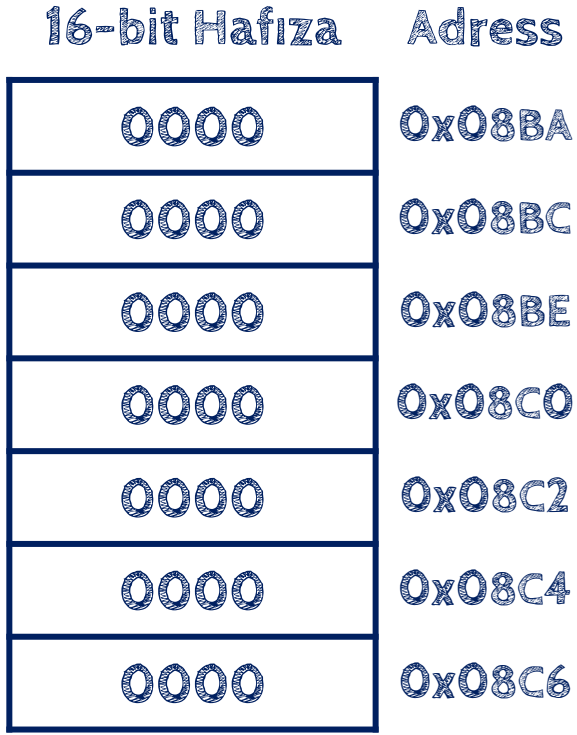
Bellek Islemleri

Suhap SAHIN
Onur GÖK

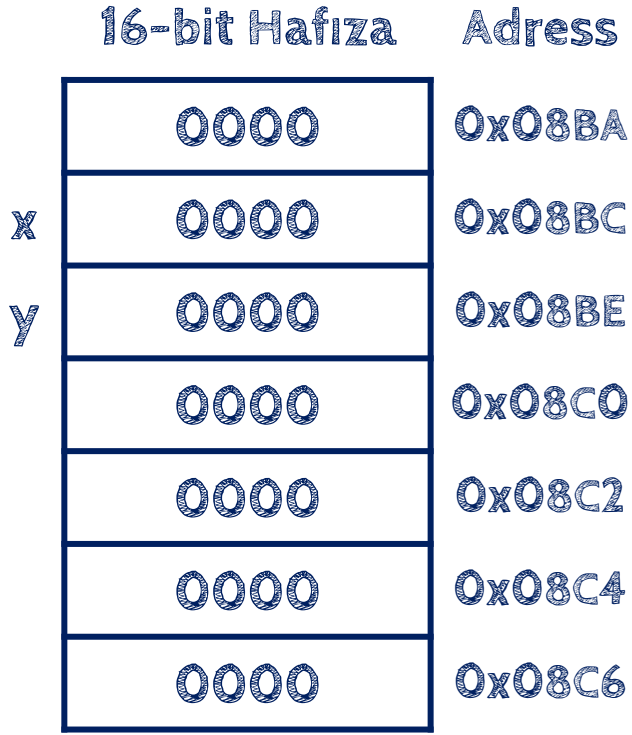
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6       x = 0xDEAD;
 7       y = 0xBEEF;
 8       p = &x;
 9      *p = 0x100;
10       p = &y;
11      *p = 0x200;
12      return 0;
13  }
```



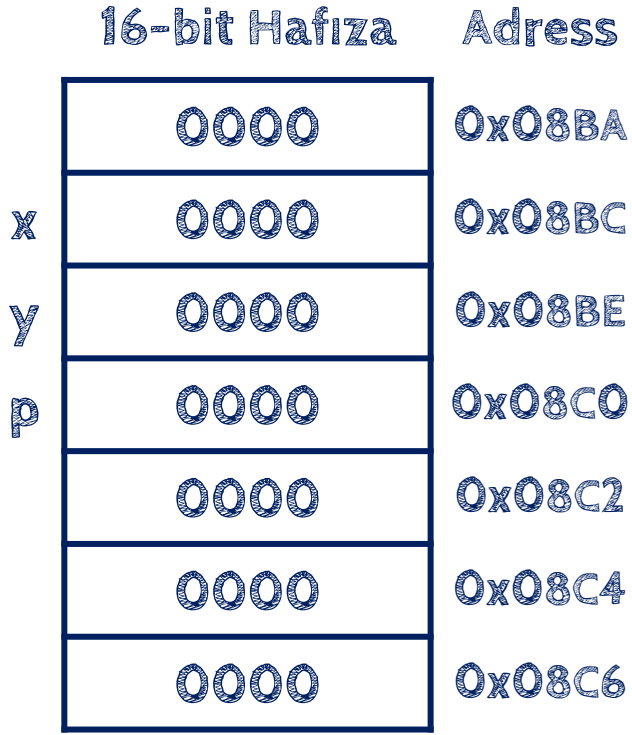
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6       x = 0xDEAD;
 7       y = 0xBEEF;
 8       p = &x;
 9      *p = 0x100;
10       p = &y;
11      *p = 0x200;
12      return 0;
13  }
```



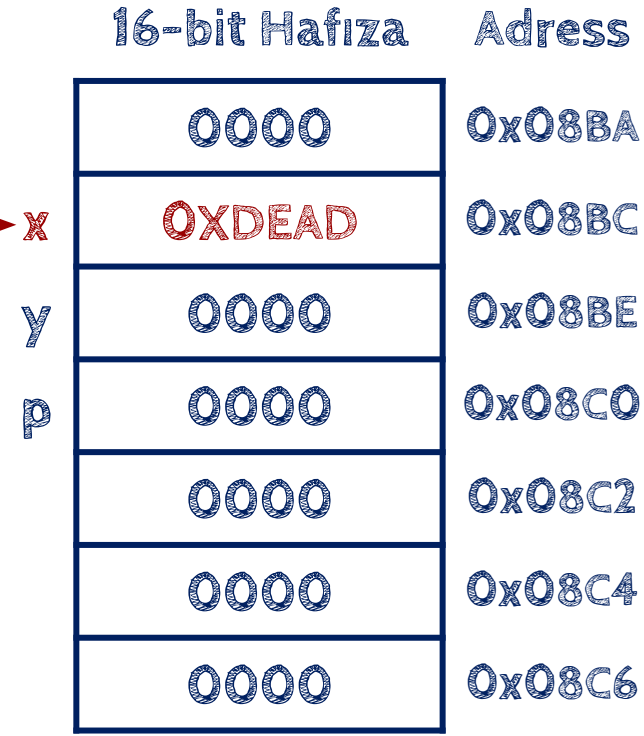
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6       x = 0xDEAD;
 7       y = 0xBEEF;
 8       p = &x;
 9      *p = 0x100;
10       p = &y;
11      *p = 0x200;
12      return 0;
13  }
```



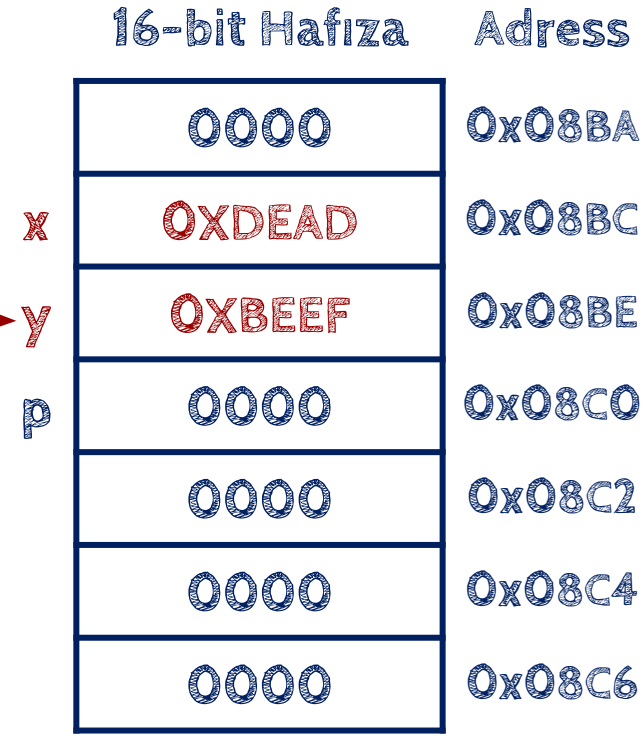
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6      x = 0xDEAD;
 7      y = 0xBEEF;
 8      p = &x;
 9     *p = 0x100;
10      p = &y;
11     *p = 0x200;
12     return 0;
13  }
```



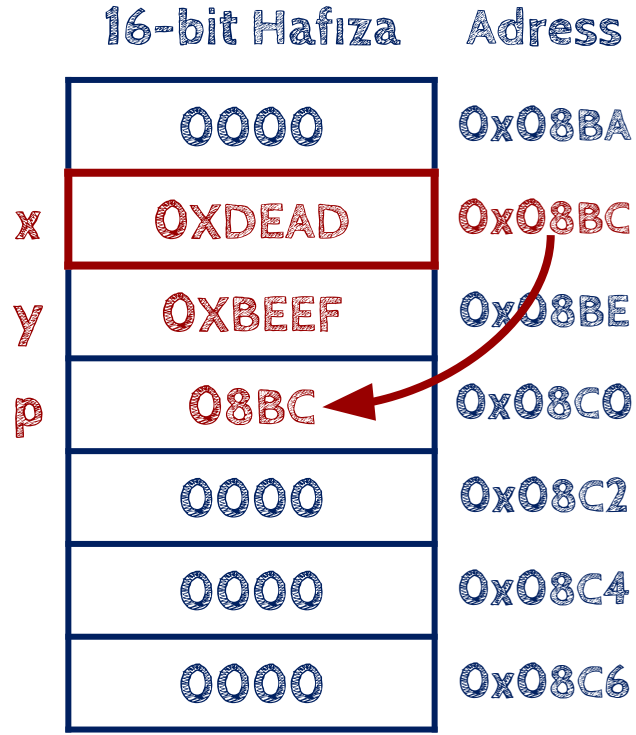
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6       x = 0xDEAD;
 7       y = 0xBEEF;
 8       p = &x;
 9      *p = 0x100;
10       p = &y;
11      *p = 0x200;
12      return 0;
13  }
```



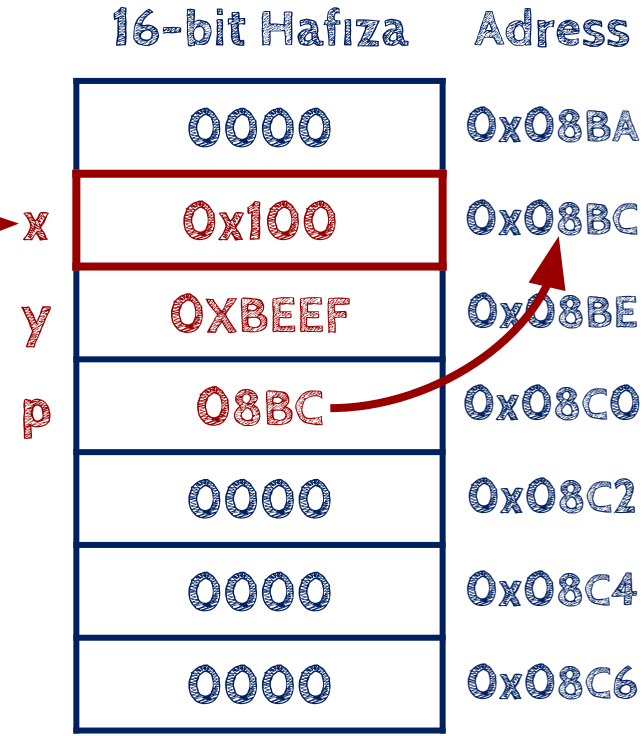
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6       x = 0xDEAD;
 7       y = 0xBEEF;
 8       p = &x;
 9      *p = 0x100;
10       p = &y;
11      *p = 0x200;
12      return 0;
13  }
```



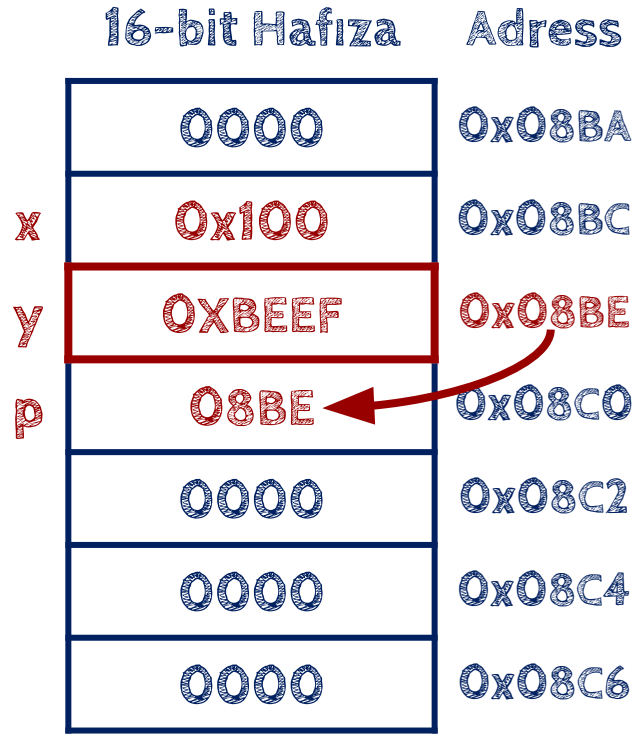
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6       x = 0xDEAD;
 7       y = 0xBEEF;
 8       p = &x;
 9     *p = 0x100;
10       p = &y;
11     *p = 0x200;
12     return 0;
13  }
```



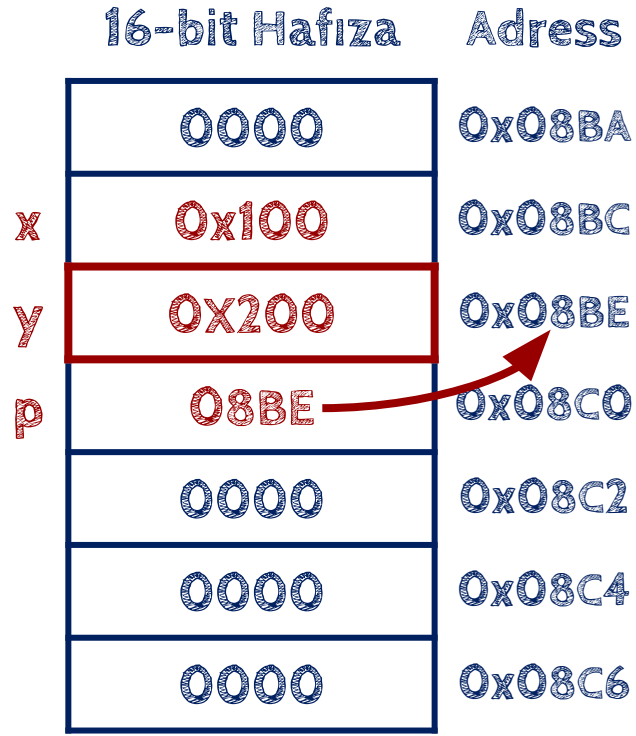
işaretçiler (Pointers)

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int x,y;
5      int *p;
6       x = 0xDEAD;
7       y = 0xBEEF;
8       p = &x;
9      *p = 0x100;
10      p = &y;
11     *p = 0x200;
12     return 0;
13 }
```



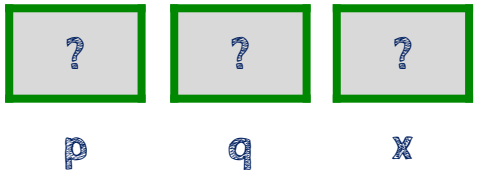
işaretçiler (Pointers)

```
 1  #include <stdio.h>
 2  int main(void)
 3  {
 4      int x,y;
 5      int *p;
 6       x = 0xDEAD;
 7       y = 0xBEEF;
 8       p = &x;
 9      *p = 0x100;
10       p = &y;
11      *p = 0x200;
12      return 0;
13  }
```

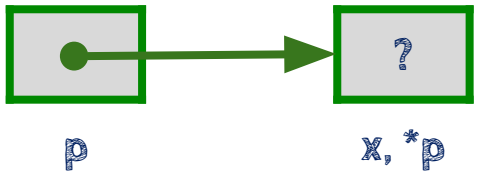


işaretçiler (Pointers)

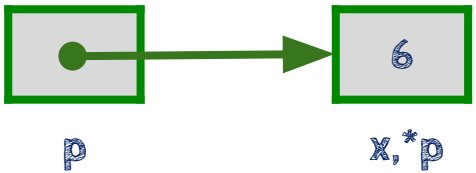
```
int *p,*q;
int x;
```



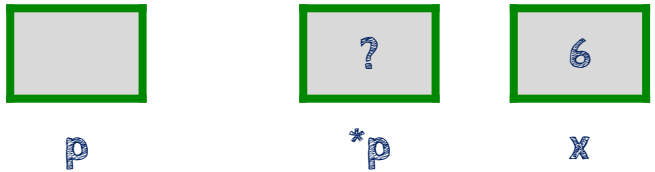

```
p = &x;
```



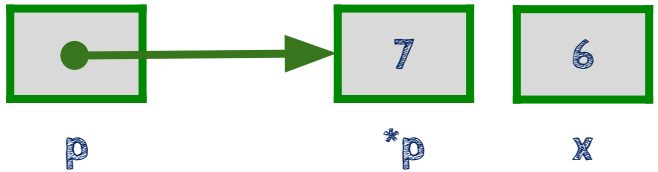

```
*p = 6;
```




```
p = new int;
```

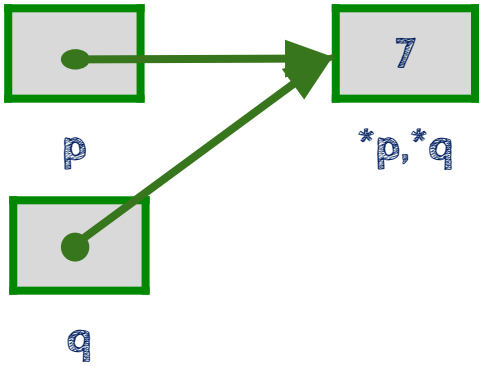



```
*p = 7;
```

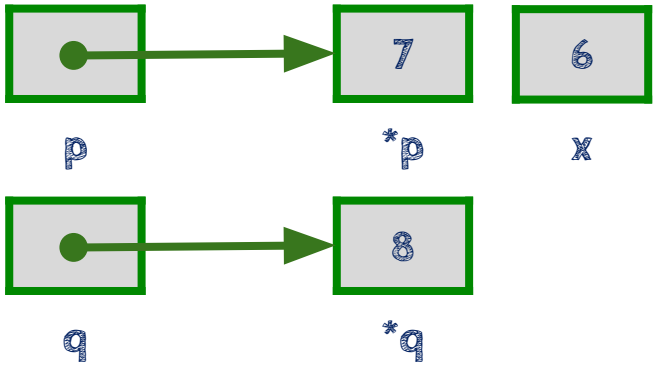


işaretçiler (Pointers)

q = p;

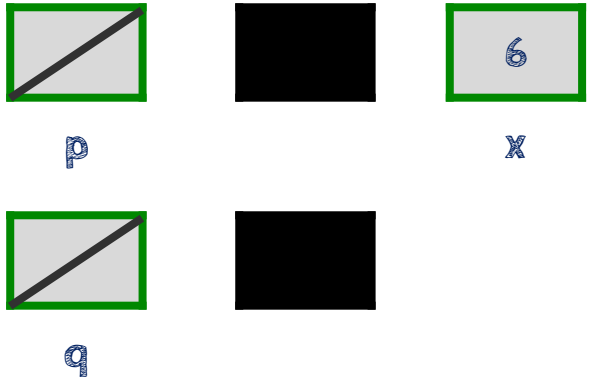


q = **new** int;
*q = 8;



p = NULL;

Delete q;
q = NULL;



Degiskenlerde Tür Dönüsümleri

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

int main() {

	float a = 3.5;


	printf("a: %f\n", a);


	return 0;
}
```

Degiskenlerde Tür Dönüsümleri

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

int main() {

	float a = 3.5;
	int b = (int)a;

	printf("a: %f\n", a);


	return 0;
}
```

Degiskenlerde Tür Dönüsümleri

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

int main() {

	float a = 3.5;
	int b = (int)a;

	printf("a: %f\n", a);
	printf("b: %d\n\n", b);

	return 0;
}
```

Verilerin Bit İfadesi

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

int main() {

	int hesap = (256*256*256)*'t' + (256*256)*'s' + 256*'e' + 't';
	printf("%d\n", hesap);

	return 0;
}
```

Isaretcilerde Tür Dönüsümleri

```c
#include <stdio.h>
#include <string.h>
int main() {
	int i;

	int dizi[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
	printf("dizi: ");
	for (i = 0 ; i < 10 ; i++)
		printf("%d ", dizi[i]);
	printf("\n\n");









	return 0;
}
```

Isaretcilerde Tür Dönüsümleri

```c
#include <stdio.h>
#include <string.h>
int main() {
	int i;

	int dizi[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
	printf("dizi: ");
	for (i = 0 ; i < 10 ; i++)
		printf("%d ", dizi[i]);
	printf("\n\n");

	char *s = (char*)(dizi);
	strcpy(s, "test");
	printf("s: %s\n\n", s);









	return 0;
}
```

Işaretçilerde Tür Dönüşümleri

```c
#include <stdio.h>
#include <string.h>
int main() {
	int i;

	int dizi[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
	printf("dizi: ");
	for (i = 0 ; i < 10 ; i++)
		printf("%d ", dizi[i]);
	printf("\n\n");

	char *s = (char*)(dizi);
	strcpy(s, "test");
	printf("s: %s\n\n", s);

	printf("dizi: ");
	for (i = 0 ; i < 10 ; i++)
		printf("%d ", dizi[i]);
	printf("\n\n");

	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

        int *A;









        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

        int *A;

        A = (int*) malloc( sizeof(int) );

        printf("A'nin gosterdigi adres: %p\n\n", A);




        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

	int *A;

	A = (int*) malloc( sizeof(int) );

	printf("A'nin gosterdigi adres: %p\n\n", A);

	*A = 123;
	printf("A'nin degeri: %d\n\n", *A);



	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

	int *A;

	A = (int*) malloc( sizeof(int) );

	printf("A'nin gosterdigi adres: %p\n\n", A);

	*A = 123;
	printf("A'nin degeri: %d\n\n", *A);

	free(A);

	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

        int *A;
        A = (int*) malloc(10 * sizeof(int));









        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

	int *A;
	A = (int*) malloc(10 * sizeof(int));

	if (A == NULL) {
		printf("HATA: bellek ayrilamadi\n");
		exit(1); // programi sonlandir
	}









	free(A);
	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

	int *A;
	A = (int*) malloc(10 * sizeof(int));

	if (A == NULL) {
		printf("HATA: bellek ayrilamadi\n");
		exit(1); // programi sonlandir
	}
	A[0] = 123;


	printf("%d\n", A[0]); // *A



	free(A);
	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {

	int *A;
	A = (int*) malloc(10 * sizeof(int));

	if (A == NULL) {
		printf("HATA: bellek ayrilamadi\n");
		exit(1); // programi sonlandir
	}
	A[0] = 123;
	A[1] = 444;
	A[9] = 674;
	printf("%d\n", A[0]); // *A
	printf("%d\n", A[1]); // *(A+1)
	printf("%d\n", A[9]); // *(A+9)

	free(A);
	return 0;
}
```

malloc & free

```
#include <stdio.h>
#include <stdlib.h>
int main() {
        int i,N,*dizi_ptr;
        printf("eleman sayisini girin: ");
        scanf("%d", &N);









        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
        int i,N,*dizi_ptr;
        printf("eleman sayisini girin: ");
        scanf("%d", &N);
        dizi_ptr = (int*) malloc( N * sizeof(int) );
        if (dizi_ptr == NULL) {
                printf("HATA: bellek ayrilamadi\n");
                exit(1); // programi sonlandir
        }









        free(dizi_ptr);
        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
	int i,N,*dizi_ptr;
	printf("eleman sayisini girin: ");
	scanf("%d", &N);
	dizi_ptr = (int*) malloc( N * sizeof(int) );
	if (dizi_ptr == NULL) {
		printf("HATA: bellek ayrilamadi\n");
		exit(1); // programi sonlandir
	}
	for (i = 0 ; i < N ; i++) {
		printf("sayi girin: ");
		scanf("%d", &dizi_ptr[i]);
	}



	free(dizi_ptr);
	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
	int i,N,*dizi_ptr;
	printf("eleman sayisini girin: ");
	scanf("%d", &N);
	dizi_ptr = (int*) malloc( N * sizeof(int) );
	if (dizi_ptr == NULL) {
		printf("HATA: bellek ayrilamadi\n");
		exit(1); // programi sonlandir
	}
	for (i = 0 ; i < N ; i++) {
		printf("sayi girin: ");
		scanf("%d", &dizi_ptr[i]);
	}
	printf("girilen sayilar:\n");
	for (i = 0 ; i < N ; i++)
		printf("%d\n", dizi_ptr[i]);
	free(dizi_ptr);
	return 0;
}
```

malloc & free

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

int main() {



        while (1) {
                int *a = (int*) malloc(100000);
        }

        return 0;
}
```

malloc & free

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

int main() {

        /* sonsuz dongude serbest birakmadan malloc fonksiyonunu kullanmak
           UYARI: asagidaki kod calisitrilirsa bilgisayar kilitlenebilir
        */
         while (1) {
                int *a = (int*) malloc(100000);
        }

        return 0;
}
```

malloc & free

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

int main() {
        int i;




        int *b = (int*) malloc(1000*sizeof(int));
        int x = 10;


        return 0;
}
```

malloc & free

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

int main() {
        int i;




        int *b = (int*) malloc(1000*sizeof(int));
        int x = 10;
        b = &x;

        return 0;
}
```

malloc & free

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

int main() {
        int i;

        /* ayrilmis alani gosteren pointera baska deger atayip alanin adresini
           kaybetmek. Programin git gide daha fazla bellek kullanmasina
           (memory leak) sebep olur. Bellek doldugunda bilgisayarin
           kilitlenmesine sebep olabilir.
         */
        int *b = (int*) malloc(1000*sizeof(int));
        int x = 10;
        b = &x;

        return 0;
}
```

malloc & free

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

int main() {
        int i;



        for (i = 0 ; i < 100000000 ; i++) {



        }

        return 0;
}
```

malloc & free

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

int main() {
        int i;



        for (i = 0 ; i < 100000000 ; i++) {
                int *c = (int*) malloc(1000*sizeof(int));
                int x = 10;
                c = &x;
        }

        return 0;
}
```

malloc & free

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

int main() {
	int i;

	/* Bir onceki hata buyuk bir dongude ise kilitlenmeye sebep olabilir.
	   UYARI: asagidaki kod calistirilirsa bilgisayar kilitlenebilir
	 */
	for (i = 0 ; i < 100000000 ; i++) {
		int *c = (int*) malloc(1000*sizeof(int));
		int x = 10;
		c = &x;
	}

	return 0;
}
```

malloc & free

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

int main() {

        int *d = (int*) malloc(1000*sizeof(int));

        return 0;
}
```

malloc & free

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

int main() {


        int *d = (int*) malloc(1000*sizeof(int));
        free(d);


        return 0;
}
```

malloc & free

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

int main() {

        int *d = (int*) malloc(1000*sizeof(int));
        free(d);
        free(d);

        return 0;
}
```

malloc & free

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

int main() {



        int *d = (int*) malloc(1000*sizeof(int));
        free(d);
        free(d);
        free(d);

        return 0;
}
```

malloc & free

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

int main() {

        /* ayni alani birden fazla kere serbest birakmak
           programin sonlanmasina veya beklenmeyen bir davranis
      sergilemesine sebep olur
        */
        int *d = (int*) malloc(1000*sizeof(int));
        free(d);
        free(d);
        free(d);

        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
        char *p_dizi;
        p_dizi = malloc(5 * sizeof(char));









        free(p_dizi);
        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
        char *p_dizi;
        p_dizi = malloc(5 * sizeof(char));
        strcpy(p_dizi, "test");
        printf("%s\n\n", p_dizi);









        free(p_dizi);
        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
        char *p_dizi;
        p_dizi = malloc(5 * sizeof(char));
        strcpy(p_dizi, "test");
        printf("%s\n\n", p_dizi);

        p_dizi = realloc(p_dizi, 100 * sizeof(char));









        free(p_dizi);
        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
        char *p_dizi;
        p_dizi = malloc(5 * sizeof(char));
        strcpy(p_dizi, "test");
        printf("%s\n\n", p_dizi);

        p_dizi = realloc(p_dizi, 100 * sizeof(char));
        strcat(p_dizi, " 123456789123456789");









        free(p_dizi);
        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
        char *p_dizi;
        p_dizi = malloc(5 * sizeof(char));
        strcpy(p_dizi, "test");
        printf("%s\n\n", p_dizi);

        p_dizi = realloc(p_dizi, 100 * sizeof(char));
        strcat(p_dizi, " 123456789123456789");
        printf("%s\n", p_dizi);
        printf("stringin boyutu: %d\n", strlen(p_dizi));
        printf("bellegin boyutu: 100\n\n");









        free(p_dizi);
        return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
	char *p_dizi;
	p_dizi = malloc(5 * sizeof(char));
	strcpy(p_dizi, "test");
	printf("%s\n\n", p_dizi);

	p_dizi = realloc(p_dizi, 100 * sizeof(char));
	strcat(p_dizi, " 123456789123456789");
	printf("%s\n", p_dizi);
	printf("stringin boyutu: %d\n", strlen(p_dizi));
	printf("bellegin boyutu: 100\n\n");

	int karakter_sayisi = strlen(p_dizi)+1; // +1 sonlandirma karakteri



	free(p_dizi);
	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
	char *p_dizi;
	p_dizi = malloc(5 * sizeof(char));
	strcpy(p_dizi, "test");
	printf("%s\n\n", p_dizi);

	p_dizi = realloc(p_dizi, 100 * sizeof(char));
	strcat(p_dizi, " 123456789123456789");
	printf("%s\n", p_dizi);
	printf("stringin boyutu: %d\n", strlen(p_dizi));
	printf("bellegin boyutu: 100\n\n");

	int karakter_sayisi = strlen(p_dizi)+1; // +1 sonlandirma karakteri
	p_dizi = realloc(p_dizi, karakter_sayisi * sizeof(char) );



	free(p_dizi);
	return 0;
}
```

malloc & free

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
        char *p_dizi;
        p_dizi = malloc(5 * sizeof(char));
        strcpy(p_dizi, "test");
        printf("%s\n\n", p_dizi);

        p_dizi = realloc(p_dizi, 100 * sizeof(char));
        strcat(p_dizi, " 123456789123456789");
        printf("%s\n", p_dizi);
        printf("stringin boyutu: %d\n", strlen(p_dizi));
        printf("bellegin boyutu: 100\n\n");

        int karakter_sayisi = strlen(p_dizi)+1; // +1 sonlandirma karakteri
        p_dizi = realloc(p_dizi, karakter_sayisi * sizeof(char) );
        printf("%s\n", p_dizi);
        printf("stringin boyutu: %d\n", strlen(p_dizi));
        printf("bellegin boyutu: %d\n", strlen(p_dizi)+1);
        free(p_dizi);
        return 0;
}
```

Sorular