```
gdb sample #Çalışabilir dosyanın sembollerini yükler. Dosyanın -g swicth ile derlenmesi gerekir

#gdb içerisinde daha sonra başka bir dosya yüklemek için
file mample

#gdb içerisinde bash komutu çalıştırmak için shell komutu kullanılır
shell clear

#Programın komut satırı argümanları set args komutu ile verilir

set args ali veli selami

#r veya run ile program çalştırılabilir. run programda breakpoint görmezse durmaz.
#start ile program çalıştırılır. main fonksiyonuna geçici breakpoint konulmuş kabul edilir. Yani main de
akış durur

#c veya continue ile break point de duran program devam ettirilebilir
#Kalıcı breakpoint için b veya break kullanılır
#Geçici breakpoint için tb veya tbreak kullanılır
#breakpoint şu şekillerde koyulabilir:

#b nin doğrudan kullanımı program çalışıyor ise bir sonraki adıma koyar
#b fonksiyon ismi o dosyadaki o fonksiyona breakpoint koyar. Şüphesiz C++ da overloading olabilir.
#b sample.cpp:foo durumunda ilgili modüldeki foo ya konur.
#b satır numarası ile ilgili satır numarasına konabilir
#b sample.cpp:10 sample.cpp deki satıra konur
#b *0x12345: adrese break point konur

#Tüm breakpoint ler delete komutu ile silinebilir
#Bir processe signal göndermek için signal komutu kullanılır
#call komutu ile process içerisinde çağrılabilen (genelde sistem fonksiyonları) fonksiyonlar çağrılabilir.


#breakpoint ler hakkında bilgi almak için info breakpoint komutu kullanılır

#kill konutu ile process sonlandırılabilir
#step (s) komutu ile tek bir satır çalıştırılabilir. Fonksiyon içine girer (step into)
#next (n) komutu yine tek bir çalıştırır. Fonksiyon içine girmez

#watch komutu ile herhangi bir nesne izlenebilir.

#C++ da bir fonksiyona breakpoint koynak için fully qualified ismi verilmelidir. Örneğin: b
csd::Sample::foo(double)

#b ile koşullu break yapılabilir. Aynı durum watch için de geçerlidir


#watch işleminde iki şekil kullanılabilir.

#watch val

#watch *p

#diziler de watch işlemiyle izlenebilir
#watch a[i]

#içiçe aynı isimli değişkenlerde kapsayan bloktaki ismin takibi * yapılması önerilir. Örneğin:

#include <iostream>

using namespace std;

int val;

int main(int argc, char **argv)
{
    int val;

    int *p = &val; // Val içteki blokta izlenecekse * izlemesi yapılmalı

    {
      int val;

      val = 40;
```

```
      *p = 30;

      cout << val << endl; //
    }

    cout << val << endl;


    return 0;
}

#analiz ediniz:

#include <iostream>
#include <cstring>
#include <cstdlib>
#include <ctime>

using namespace std;

const char g_text[] = "0123456789abcdef";
const int g_len = 13;

void overrun(char *buf, const char *msg, int len)
{
  char dummy[4096];

  memset(dummy, len, sizeof(dummy));

  memcpy(buf, msg, len);
}

int main(int argc, char **argv)
{
    char *buf = new char[g_len];
    int i;

    srand(static_cast<unsigned int>(time(0)));

    int n = 1000;

    int thresh = RAND_MAX / n;

    for (i = 0; i < n; ++i) {
      int len = rand() < thresh ? g_len + 1: g_len;
      overrun(buf, g_text, len);
    }

    int overran = strlen(buf) > g_len;

    if (overran)
      cout << "overrun" << endl;
    else
      cout << "No overrun" << endl;


    delete buf;


    return 0;
}

#set var komutu ile bir değişkenin debug aşamasında içeriği değiştirilebilir

#bt veya backtrace komutu ile çağrıma stack inin komutuna bakılır.

#info locals komutu o an (context) bulunan yerel değişkenleri gösterir
#whatis komutu bir sembolün hakkında bilgi verir

#printf(p) komutu C de printf gibi düşünülebilir. argüman olarak aldığı nesnenin bilgisini terminale verir
#examine komutu bir adres için kullanılabilir

#x için bir takım özel durumlar vardır. Dökümanlar incelenebilir

#template fonksiyonlar için (template sınıfların fonksiyonu) hangi açılım için break point konulacaksa tam
```

```
#template fonksiyonlar için (template sınıfların fonksiyonu) hangi açılım için break point konulacaksa tam ismiyle verilmelidir

#include <iostream>
#include <cstring>
#include <cstdlib>
#include <ctime>

using namespace std;

template <class T>
void foo(T t)
{
    cout << t <<endl;
}


int main()
{
    foo(10);
    foo(2.3);

    return 0;
}

#b foo<int>
#b foo<double>

#Koşullu watch komutları da yazılabilir:
#include <iostream>

using namespace std;

int main()
{
    int a;


    cout << "Bir sayı giriniz" << endl;
    cin >> a;

    a *= 3;

    return 0;
}

#watch a if a == 30
```