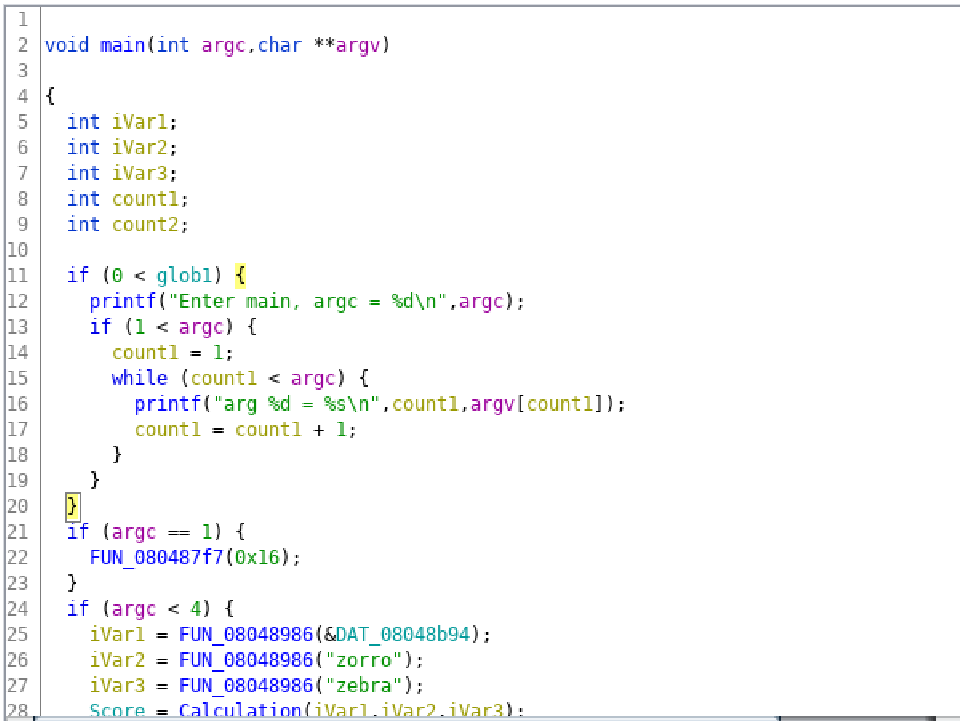
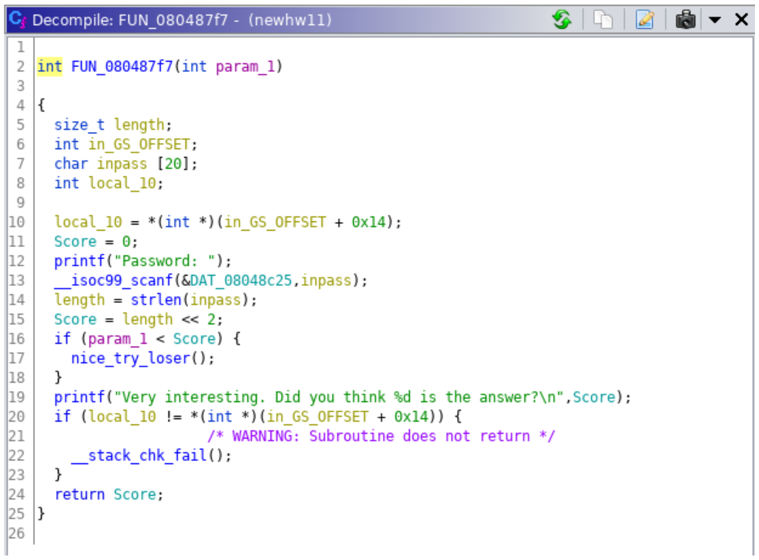


```
void main(int argc,char **argv)

{
  int iVar1;
  int iVar2;
  int iVar3;
  int count1;
  int count2;

  if (0 < glob1) {
    printf("Enter main, argc = %d\n",argc);
    if (1 < argc) {
      count1 = 1;
      while (count1 < argc) {
        printf("arg %d = %s\n",count1,argv[count1]);
        count1 = count1 + 1;
      }
    }
  }
  if (argc == 1) {
    FUN_080487f7(0x16);
  }
  if (argc < 4) {
    iVar1 = FUN_08048986(&DAT_08048b94);
    iVar2 = FUN_08048986("zorro");
    iVar3 = FUN_08048986("zebra");
    Score = Calculation(iVar1,iVar2,iVar3);
```

The first thing I did was to load the program into ghidra and to look at how the program was laid out.

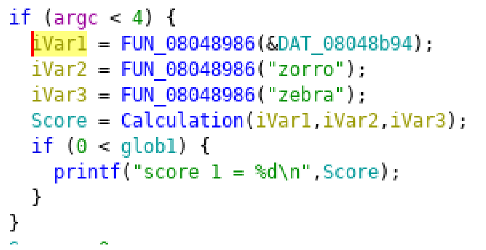
There was no main function but the entry did call a function which looked like main. After class on Friday And we were given argc and **argv it made it a little more readable and you could see how the program worked a little better.



Decompile: FUN_080487f7 - (newhw11)

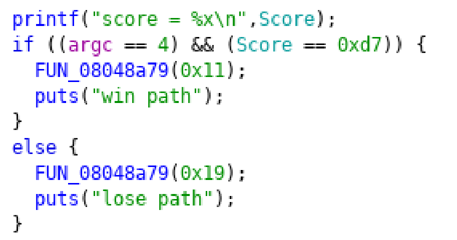
```
int FUN_080487f7(int param_1)

{
  size_t length;
  int in_GS_OFFSET;
  char inpass [20];
  int local_10;

  local_10 = *(int *)(in_GS_OFFSET + 0x14);
  Score = 0;
  printf("Password: ");
  __isoc99_scanf(&DAT_08048c25,inpass);
  length = strlen(inpass);
  Score = length << 2;
  if (param_1 < Score) {
    nice_try_loser();
  }
  printf("Very interesting. Did you think %d is the answer?\n",Score);
  if (local_10 != *(int *)(in_GS_OFFSET + 0x14)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return Score;
}
```

This is the function that is called if you pass no arguments to it when you execute it. After running the program and looking at this you can see that is no path to the win so its just a way to get users thinking you must input a password.



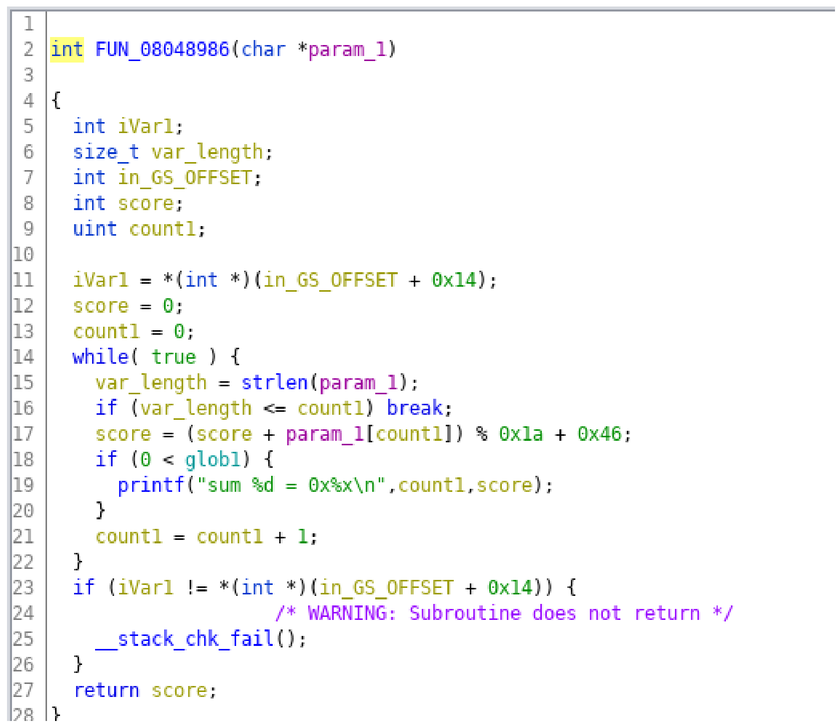
```
if (argc < 4) {
  iVar1 = FUN_08048986(&DAT_08048b94);
  iVar2 = FUN_08048986("zorro");
  iVar3 = FUN_08048986("zebra");
  Score = Calculation(iVar1,iVar2,iVar3);
  if (0 < globl) {
    printf("score 1 = %d\n",Score);
  }
}
```

This was the thing that caught my attention first. I noticed that argc must be less then 4 if you want to execute this block of code, so the program must take 4 arguments when executing. It also calls the same function for the 3 arguments and then takes them to another function that calculates the score and stores it in the Score Global.



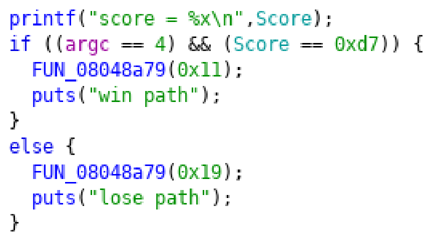
```
printf("score = %x\n",Score);
if ((argc == 4) && (Score == 0xd7)) {
  FUN_08048a79(0x11);
  puts("win path");
}
else {
  FUN_08048a79(0x19);
  puts("lose path");
}
```

This also caught my eye. I know if this was anything like the homework we did last week then the answer is usually found with a compare of sum sort. This also has the Score variable set to d7 which is 215 in decimal.



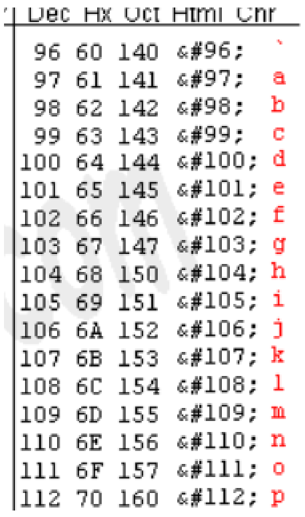
```

int FUN_08048986(char *param_1)

{
  int iVar1;
  size_t var_length;
  int in_GS_OFFSET;
  int score;
  uint count1;

  iVar1 = *(int *)(in_GS_OFFSET + 0x14);
  score = 0;
  count1 = 0;
  while( true ) {
    var_length = strlen(param_1);
    if (var_length <= count1) break;
    score = (score + param_1[count1]) % 0x1a + 0x46;
    if (0 < globl) {
      printf("sum %d = 0x%x\n",count1,score);
    }
    count1 = count1 + 1;
  }
  if (iVar1 != *(int *)(in_GS_OFFSET + 0x14)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return score;
}
```

This is the function that calculates the score from the arguments passed to the program. This function iterates through the string passed too it character by character and adds it to the score variable. It first takes the ascii value of the argument passes into it and mods it by 26 (1a in hex) and then adds 70 (46 in hex). This is then returned to the main function.



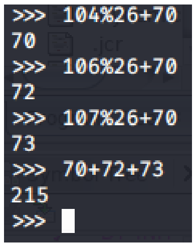
```
printf("score = %x\n",Score);
if ((argc == 4) && (Score == 0xd7)) {
  FUN_08048a79(0x11);
  puts("win path");
}
else {
  FUN_08048a79(0x19);
  puts("lose path");
}
```

If we look again at this part of the code we know that score must equal d7 which is 215 in decimal. So based off how the score is calculated we need 3 characters that will equal 215. I knew that if we needed score to equal 215, I didn't think that this would be very hard, I was worng and right. I went to an ascii table and started looking for characters. After about 5 different combinations I realized that the easiest way to get this would to have the first character equal 70, the next character to equal 72 and then 73 for the last character.



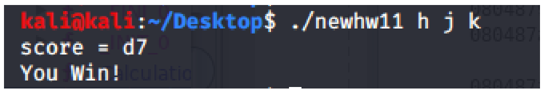
Dec	Hx	Oct	Html	Chr
96	60	140	&#96;	`
97	61	141	&#97;	a
98	62	142	&#98;	b
99	63	143	&#99;	c
100	64	144	&#100;	d
101	65	145	&#101;	e
102	66	146	&#102;	f
103	67	147	&#103;	g
104	68	150	&#104;	h
105	69	151	&#105;	i
106	6A	152	&#106;	j
107	6B	153	&#107;	k
108	6C	154	&#108;	l
109	6D	155	&#109;	m
110	6E	156	&#110;	n
111	6F	157	&#111;	o
112	70	160	&#112;	p

I looked at these ascii values but it could have been any of them .



```
>>> 104%26+70
70
>>> 106%26+70
72
>>> 107%26+70
73
>>> 70+72+73
215
>>> 
```

Since I wanted the first character to equal 70 then it had to be a factor of 26 so I started at 104 (26*4). This gave me 70, I then just looked at the table and used j since that would give me 72 and k to give me 73. This gave me 215, time to run and see if I win.



```
kali@kali:~/Desktop$ ./newhw11 h j k
score = d7
You Win!
```

Like I thought this equaled d7 and gave us the you win print.