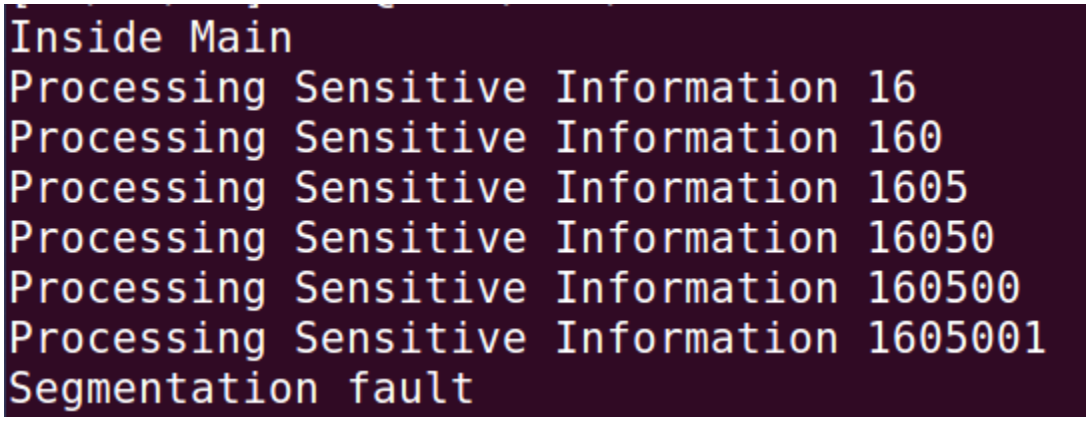
CSE 406
Online -1 (A - 2)

You are given the following vulnerable C program *A2.c*. Replace <param_1> and <param_2> in the source code with the corresponding values of Table-1.

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

int foo(int a, int b);
int bar(int x);

int bof(char *str)
{
    char buffer[<param_1>];
    /* The following statement has a buffer overflow problem */
    strcpy(buffer, str);
    //printf("Returning from BOF\n");
    return 1;
}
int foo(int a, int b)
{
    int secret = a*10 + b;
    printf("Processing Sensitive Information %d\n",secret);
    return secret;
}
int bar(int x)
{
    printf("Input Parameter %d\n",x);
    return x+1;
}

int main(int argc, char **argv)
{
    char str[<param_2> + 1];
    FILE *badfile;
    badfile = fopen("badfile", "r");
    printf("Inside Main\n");
    fread(str, sizeof(char), <param_2>, badfile);
    bof(str);

    printf("Returned Properly\n");
    return 1;
}
```

Tasks:

1. First, compile the program as shown in the lab. Do not forget to turn off address space randomization and stack protection. Also, make sure that the stack is executable while compiling the program.
2. Prepare a payload (e.g. badfile) which will cause the program to print your student ID in the following format.



3. Note that you have to repeatedly call the function `foo` to print your student ID. You have to utilize the return value of the last call for the next call.
4. Rename your ***exploit.py*** file with ***16050xx.py*** and submit in moodle.

Mark Distribution

Item	Marks
Single call to `foo` (i.e. Print 16)	7
Repeated call to `foo` (i.e. Print 16050XX)	10
Reuse of return value	3
Total	**20**

Hint

You already have the payload to open a shell. Steps to cause the overflow is similar to opening a shell. You just need to replace the payload of opening a shell with your own payload which repeatedly calls `foo.`

For example, the following payload calls the function **bar** twice.

```
shellcode_3 = (
"\x6a\x04"              #push   4             (Setting Parameter x1 = 4)
"\xbb\x3e\x85\x04\x08" #mov    ebx,0x804853e (Move entry point of bar to edx)
"\xff\xd3"              #call   ebx           (Call the function bar)

"\x50"                  #push   eax           (Return value is stored at eax.
                        #                      Setting Paratemer x1 = last return value)
"\xff\xd3"              #call   ebx           (Calling bar again)
).encode('latin-1')
```

Output of the above payload:

```
Inside Main
Input Parameter 4
Input Parameter 5
Segmentation fault
```

If you need to convert assembly to machine code follow this website:
https://defuse.ca/online-x86-assembler.htm#disassembly

Table - 1

Student ID	Param_1	Param_2
1605031	977	2197
1605032	843	1655
1605033	741	2217
1605034	921	1828
1605035	422	2375
1605036	316	2049
1605037	499	2010
1605038	300	1684
1605039	993	1679
1605040	443	1970
1605041	310	1753
1605042	366	1744
1605043	302	1638
1605044	893	2322
1605045	938	2017

1605046	415	2076
1605047	327	1606
1605048	534	1647
1605049	423	2094
1605050	503	1771
1605051	645	2244
1605052	925	2260
1605053	761	2337
1605054	743	1868
1605055	442	2195
1605056	340	1817
1605057	510	2295
1605058	693	2328
1605059	787	1706
1605060	530	1950
0905081	709	2341
1405059	975	2350
1405081	623	1975
1605013	379	1609
1605019	817	1703
1605021	799	2116
1605027	759	1933
1605065	740	2144
1605067	532	1690
1605076	734	2100
1605080	982	1984
1605085	950	1689
1605089	407	2171
1605090	421	1759