EXERCISE NO. 1

SUBMISSION DUE DATE: 6/4/2016 23:55

INTRODUCTION

The purpose of this assignment is to get familiar with Linux and development environments. The assignment includes the following:

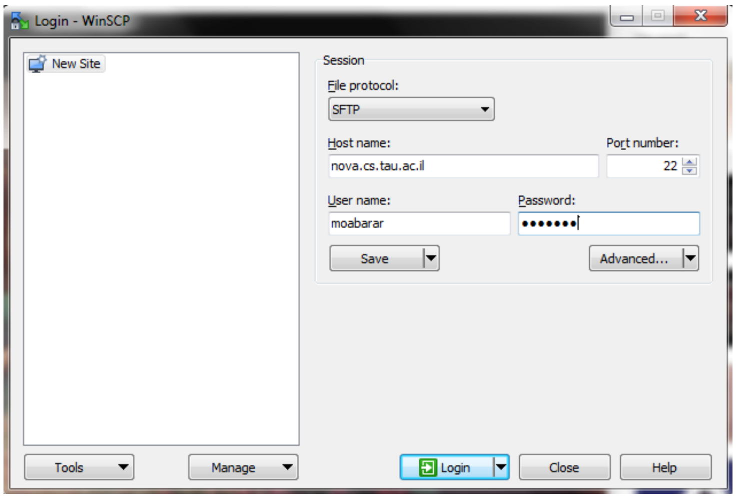
1- Remotely connect to the Nova server and get familiar with basic shell commands.
2- Create your first C Program. The program will be a simple command line program which does cool math stuff.
3- Compiling, debugging and basic use of makefiles.

DEVELOPMENT ENVIRONMENT

Students may work in any development environment they like. However we encourage you to use eclipse as your development tool (we will not support issues regarding other IDEs or operating systems other than Linux or windows). Your submission will be automatically checked by a script, so your code should run properly on Nova - the faculty server. You can use the computers in the PC farm or remotely connect to the server as will be described below.

FILE TRANSFERRING

If you need to transfer files from your personal computer to Nova, you can use WinSCP. To connect you will need to enter your personal authentications. See the example below.

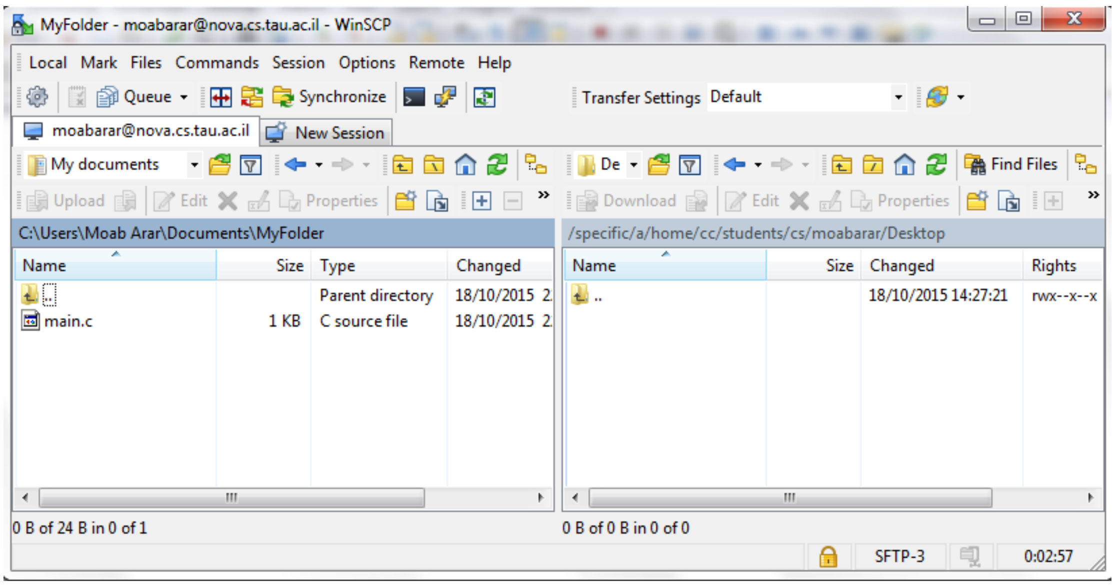


Login - WinSCP

New Site

Session
File protocol: SFTP
Host name: nova.cs.tau.ac.il
Port number: 22
User name: moabarar
Password: •••••••

After connecting you can upload your files by dragging the designated file from your personal computer to the directory in the Nova server:



MyFolder - moabarar@nova.cs.tau.ac.il - WinSCP

moabarar@nova.cs.tau.ac.il

C:\Users\Moab Arar\Documents\MyFolder

Name	Size	Type	Changed
..		Parent directory	18/10/2015 2
main.c	1 KB	C source file	18/10/2015 2

0 B of 24 B in 0 of 1

/specific/a/home/cc/students/cs/moabarar/Desktop

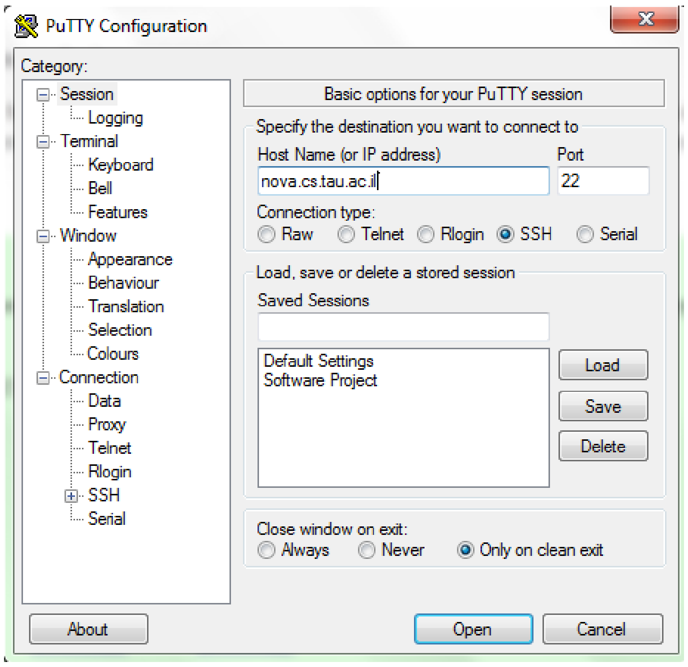
Name	Size	Changed	Rights
..		18/10/2015 14:27:21	rwx--x--x

0 B of 0 B in 0 of 0

SFTP-3 0:02:57

REMOTE CONNECTING TO NOVA

To connect to the Nova server you can use PuTTY:



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): nova.cs.tau.ac.il Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

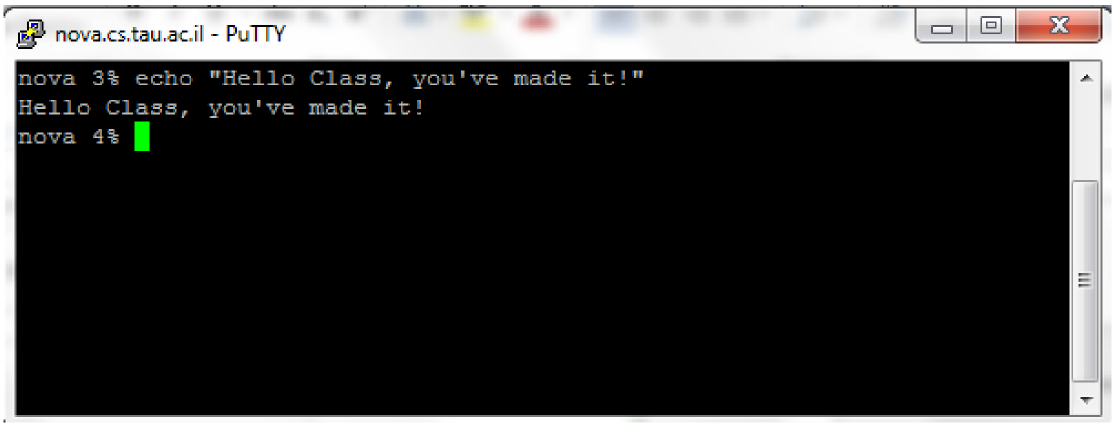
Load, save or delete a stored session

Saved Sessions

Default Settings
Software Project

Close window on exit: Always Never [✓ Only on clean exit]

After pressing open you will be asked to enter your username and password for authentication. (The username and password are the same as in your moodle authentication.) When successfully connected to Nova you will be able to see the terminal as in the following picture:



```
nova 3% echo "Hello Class, you've made it!"
Hello Class, you've made it!
nova 4%
```

Now you can enter shell commands and execute them by pressing enter. See below to find out more about shell commands.

(Note: echo is a shell command that is used to print strings to the standard output.)

USEFUL LINKS

You can download PuTTY and WinSCP by clicking on the following link. Recall that we recommend that you work with eclipse; please follow the installation guidelines for eclipse in moodle.

BASIC SHELL COMMANDS

After the connection with the server (Nova) is established, you can now type in shell commands and execute them simply by pressing enter. The results will be shown on the terminal (If there's any).

A few useful shell commands:

```
>> pwd
```

Prints the full pathname of the current working directory to the standard output. (The pathname is relative to the root directory which is the first directory in Linux.)

```
>> ls [dir]
```

Lists the content of the directory "*dir"* (Both files and directories). If no parameters are given, the result is the content of the current working directory.

```
>> cd [dir]
```

Changes the current directory to be *dir.*
Use the following shortcuts:

"." – This is a shortcut for the current directory.
".." – This is a shortcut for the parent directory in the hierarchy.
"~" – this is a shortcut for the home directory.

```
>> mkdir [dirName]
```

Creates a new directory with the name dirName.

```
>> cp [file1] [file2] … [fileK] [dir]
```

Copies *file1,file2,…,fileK* to the directory *"dir".*
Note: In order to copy an entire directory (recursively copy a directory) use –r flag.

Examples:

```
>> cp /dir1/myFile /dir2
```

Copies "*myFile*" which is located in *"/dir1"* to the directory "*/dir2*"

```
>> cp –r /sourceDir /destinationDir
```

This copies the directory "*sourceDir*" (with its content) to the directory "*destinationDir*"

```
>> rm [file1] [file2] … [fileK]
```

Removes *file1,file2,…,fileK.*

```
>>man [command]
```

The man command is used to display the manual page of the command "*command*".
Note: you can navigate through the manual page using the arrows in the keyboard. To exit the manual page press "*q*"

```
>> diff [file1] [file2]
```

Prints the difference between the two files (file1 and file2)
Note: Use the *man* command to see the manual page of the command *"sdiff".*

FIRST C PROGRAM

In this assignment you will write your first C program. The program consists of a library called **MathIsFun** (as seen in the tutorial) and a main function. The header file of the library is called MathIsFun.h, it contains the declarations of the functions we will implement alongside its documentation. You can find the header in the assignment zip file; please review it before proceeding with the instructions.

Your task is in this assignment is to implement the interface given by the header file **MathIsFun.h**. The implementation must be in the source file **MathIsFun.c**. You will also have to implement a simple main function, as will be shortly described.

As we saw in class, we think of the header files as contracts between the user and the programmer of each header file. Anyone who is going to use our library (MathIsFun.h) cannot assume anything about the behavior of our code other than what is given by the documentation. The user refers to the library as a black-box, and it's our responsibility to make sure our code is compatible with the contract (header). This principle is called information hiding and we will use it extensively throughout this course.

MATHISFUN

You will have to implement the functions declared in the header file MathIsFun.h. The implementation should be in the file MathIsFun.c. This file contains the implementation of the following functions:

1-
```c
int funPow(int x, int n, int d);
```
This function calculates the remainder of pow(x,n) divided by d ($(x^n mod\ d)$. The run-time complexity of this function is $O(\log(n))$.
Hint: Write a recursive function on **n** which calculates **pow(x,n) mod d** using the following properties:

a. If n is even: $x^n mod\ d = x^{\frac{n}{2}} \times x^{\frac{n}{2}}\ mod\ d$ – How can you extend this property if n is odd?
(**Hint:** if n is odd then n-1 is even, thus $x^{n-1} mod\ d = x^{\frac{n-1}{2}} \times x^{\frac{n-1}{2}}\ mod\ d$).

b. If $a < 0$ then $a\ mod\ d = (a + d)\ mod\ d$.
Thus the modulo value must always be in the range {0,..,d-1} (Note that the **%** operator in C on negative numbers is also negative).
For example **-1 mod 3 = 2 mod 3** (In C -1%3 = -1, but funPow(-1,1,3) = 2).

c. For any integers $a \times b\ mod\ d = ((a\ mod\ d) \times (b\ mod\ d)) mod\ d)$
Use this property to define your recursive step.

d. In this assignment we will assume that for any integer x it holds that $x^0 = 1$.
($0^0 is\ undefined$ but in our case we will assume $0^0 = 1$)

2-
```c
bool funPrimeCheck(int x);
```
This function checks if x is prime. The run-time complexity is $O(\sqrt{\mathrm{x}})$.
Hint: in order to check if x is prime, it is suffice to check if x is divided by $\{2, .., \lfloor\sqrt{x}\rfloor\}$.

3- You will need to implement the following function in order to implement funPrimeCheck. This function will not be visible to the users of our library.
```c
/*
 * Calculates the largest integer less or equal than the square root of x.
 * funSqrt(10) = 3
 * funSqrt(16) = 4
 * @param x - An integer for which the function applies
 * @return the value of |_sqrt(x)_|, if x is negative then the function
 * returns -1.
 */
int funSqrt(int x);
```
This function calculates $\lfloor\sqrt{x}\rfloor$, if x is negative then it returns -1. The run-time complexity of this function is $O(\log(x))$.
Hint: You could calculate $\lfloor\sqrt{x}\rfloor$ by finding $0 \le i \le x\ s.t\ i^2 \le x < (i + 1)^2$. Use binary search to achieve $O(\log(x))$ complexity.

4- `bool funPalindromeCheck(int x);`
This function checks if x is a palindrome, namely a number that is the same when written forwards and backwards**. Note, negative numbers are not palindromes.** Run-time complexity $O(Number\ of\ digits\ of\ x) = O(\log_{10}(x))$.

Remarks:

1- **It is important not to use any C libraries in order to implement your code. Any solution which uses C libraries will not be accepted!**

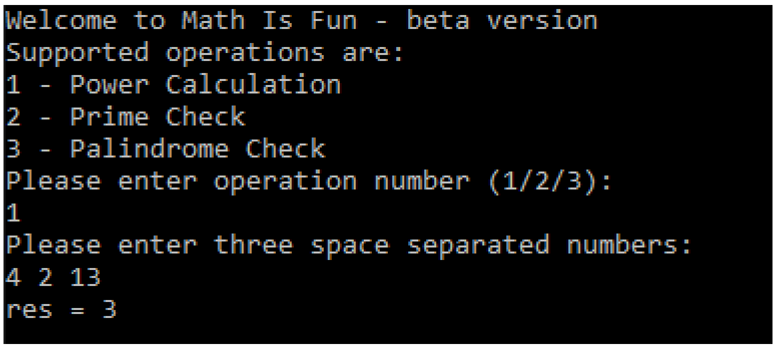
MAIN

The main function is the entry point of any C program. We will avoid implementing other functions inside the main.c file. In this assignment however, if you do wixh to implement auxiliary functions which will be used in the main function you can do that in main.c.
The behavior of the main function is as follow:

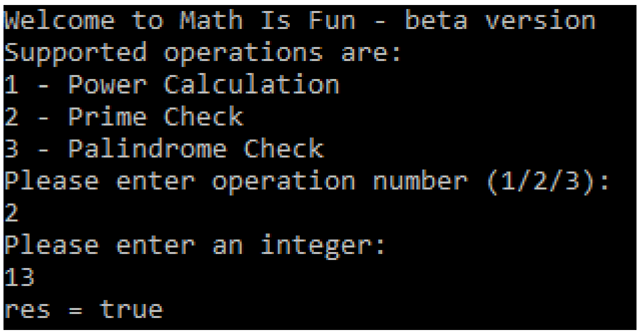
1- **<Display Instruction message 1>:**
Our program will first display an instruction message, explaining to the user what kind of operation it supports. You will need to print the following message:
"Welcome to Math Is Fun - beta version\n"
"Supported operations are:\n"
"1 - Power Calculation\n"
"2 - Prime Check\n"
"3 - Palindrome Check\n"
"Please enter operation number (1/2/3): \n"
2- **<Receive input from user>:**
The program will then receive from the user an integer representing the operation.
3- **<Interpret Operation>:**
The program will interpret the operation:
   a. [If the operation is Power calculation] :
   <Display Instruction Message 2>:
   The program will ask the user to enter three space-separated numbers you will need to first print the following message.
   "Please enter three space separated numbers: \n"
   <Receive input from user>:
   The program will receive from the user three space-separated integers
   <Display result>
   The program will display the result as follow:
   Let **val = pow(x,n,d)**, then the program will print:
   "res = val\n"
   b. [If the operation is Prime test/Palindrom test]:
   <Display Instruction Message 2>:
   The program will ask the user to enter an integer. You will need to first print the following message.
   "Please enter an integer: \n"
   <Receive input from user>:
   The program will receive from the user an integer
   <Display result>
   The program will display the result as follow:
   If x is prime/palindrome the program will display:

“res = true\n”
otherwise it will display:
“res = false\n”

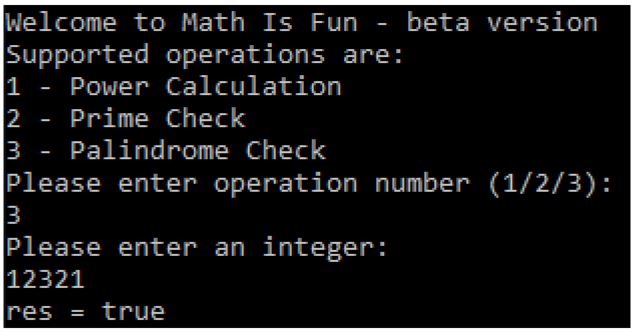
Examples:



```
Welcome to Math Is Fun - beta version
Supported operations are:
1 - Power Calculation
2 - Prime Check
3 - Palindrome Check
Please enter operation number (1/2/3):
1
Please enter three space separated numbers:
4 2 13
res = 3
```



```
Welcome to Math Is Fun - beta version
Supported operations are:
1 - Power Calculation
2 - Prime Check
3 - Palindrome Check
Please enter operation number (1/2/3):
2
Please enter an integer:
13
res = true
```



```
Welcome to Math Is Fun - beta version
Supported operations are:
1 - Power Calculation
2 - Prime Check
3 - Palindrome Check
Please enter operation number (1/2/3):
3
Please enter an integer:
12321
res = true
```

Remarks:

1- **You code will be automatically tested. Please follow the printing instruction, if you use different format, you code will fail our tests.**

COMPILE, DEBUG AND MAKEFILE

As stated before, students may work on any development environment they choose. However your code should compile and run on Nova. So in order to compile the program you wrote, please connect to Nova and copy all relevant files (main.c, MathIsFun.c, MathIsFun.h and makefile) to a directory of your choice (note that all the files should be in the same directory).

When the connection is established, set your current directory to the directory where the assignment files are located. Type in the following command and press enter in order to build your program:

```
>>   gcc -std=c99 -Wall -Wextra -Werror -pedantic-errors  main.c  MathIsFun.c-o ex1
```

The following command will compile your program and will create a binary file called ex1. Note that the compilation flags used in the command above will be our default flags, so it is recommended that you configure your eclipse to compile your code with these flags. (Follow the instructions on how to add compilation flags in eclipse : you can find these in Eclipse guide in moodle.)

Now you can run your program by executing the following line on Nova:

```
>> ./ex1
```

Another way to check your program is by using the input files that you were given in assignment1.zip. To do so we will use I/O redirection: in our case we will run the program where the standard input is from a file (for instance test1.in) and the standard output will be another file (for instance result1.out). Use the expected output to check your results.

For example to check your code you could follow these steps:

1- Compile your program.
2- Use I/O redirection as follow:
   ```
   >> ./ex1 < test1.in > result1.out
   ```
   this will use "*test1.in*" as the standard input and write the results to the file "*result1.out*" (The file will be created if it doesn't exist.)
3- Compare your results with the expected result for "*test1.in*", the expected result for "*test1.in*" is "*expected1.out*".
   ```
   >> diff result1.out expected1.out
   ```
4- If **nothing** was printed, then your results matches the results we got (Good job). If not, try to debug and find the source of your mistake.
5- You can repeat steps 1-4 for test2.in, test3.in etc…

You can also use the makefile provided to compile your code. Make sure your code compiles with the makefile you were given, to do so please follow these guidelines:

1- Copy your source code along with the file "makefile" into the Nova server. Place the files in the same directory. (IF you have already done so in previous round, just ignore this)
2- Go the directory where you copied the files to, and simply write the following command:
   ```
   >> make
   ```
3- As an output, make sure the files "main.o" , "MathIsFun.o" and "ex1" were added to the directory (use "ls" command). You can run your program by executing the file "ex1" as stated in previous sections.

Note: Please look at the makefile and read the guidelines in the course material for more information on makefile. You will need to implement one on your own in future assignments.

SUBMISSION

Please submit a zip file named **id1_id2_assignment1.zip** where id1 and id2 are the ids of the partners. The zipped file must contain the following files:

- main.c – Your source code for the program.
- MathIsFun.h – the header file of MathIsFun.h (This file should be the same as in the assignment zip file – Don't Change it!)

- MathIsFun.c – Your implementation for the library MathIsFun.
- partners.txt – This file must contain the full name, id and moodle username for both partners. Please follow the pattern in the assignment files. **(do not change the pattern)**
- makefile – the makefile provided in the assignment files. (Don't change it!)

Note: Both students should submit their code!

REMARKS

- For any question regarding the assignment, please don't hesitate to contact Moab Arar by mail: moabarar@mail.tau.ac.il.
- Late submissions are not acceptable unless you have the lecturer approval.
- Borrowing from others' work is not acceptable and may bear severe consequences.

GOOD LUCK