Data Structures and Algorithms Assignment

CSA1017
Data Structures and Algorithms 1

Dr. John Abela
Mr. Kristian Guillaumier

Linux Mint Debian Edition

Stefan Mallia
0128591m
Bachelor of Science (Honours)
Computing Science
and
Statistics and Operations Research

University of Malta
Department of Intelligent Computer Systems

Table of Contents

Table of figures

Question 1: Decimal Numbers to Roman Numerals

1.1: The dec_to_rom() function

A new function is declared that takes an unsigned int as input and returns a string. The function starts by declaring a new struct, Rom_node, that contains two types. The first is an *unsigned int* and the second is a *char const*. An array of these structs is then created to associate all roman numeral symbols with the respective decimal number.

Using the above array and the function input, a while loop is performed to create the roman numeral string. The while loop starts by using the first element in the struct array, that is, {1000, “M”}. This can be referred to as the current struct element and is identified by the program by using an index variable, roman_index.

With each loop, if the input is found to be greater or equal to the current struct array element, then the symbol of that element is appended to the string result and the input is decreased by the number of the struct array element.

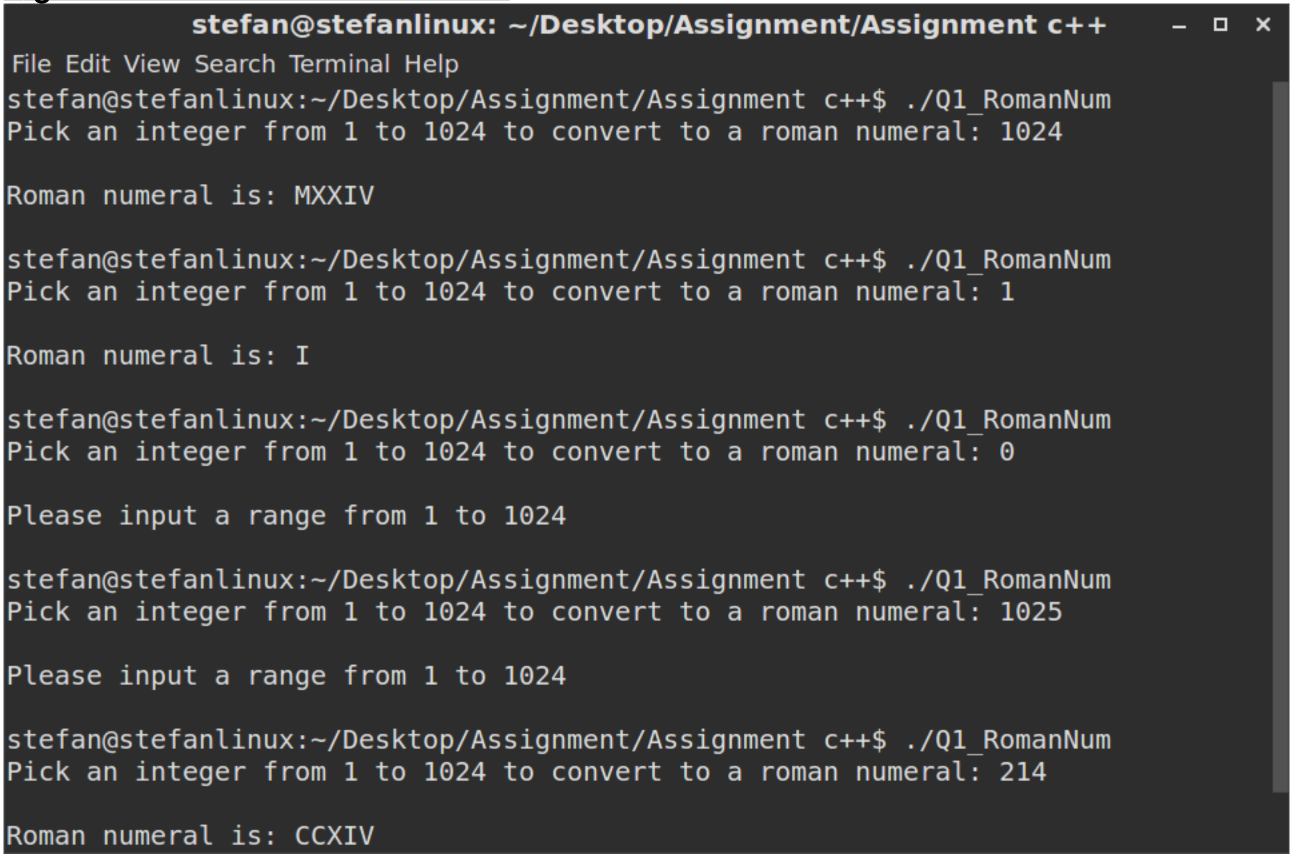
If the input is not found to be greater or equal to the current struct array element then the current struct element is changed to the next. So if the input is not found to be greater or equal to 1000 from the {1000, “M”} element, the current struct element becomes {900, “CM”}.

This process continues until the inputted value is decreased to 0 or below 0.

1.2: Testing

The main function implements the dec_to_rom() function by requesting that the user input a number from 1 to 1024, although the dec_to_rom() is capable of processing a wider range of values.

<u>Figure 1.1: Decimal to roman test</u>


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q1_RomanNum
Pick an integer from 1 to 1024 to convert to a roman numeral: 1024

Roman numeral is: MXXIV

stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q1_RomanNum
Pick an integer from 1 to 1024 to convert to a roman numeral: 1

Roman numeral is: I

stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q1_RomanNum
Pick an integer from 1 to 1024 to convert to a roman numeral: 0

Please input a range from 1 to 1024

stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q1_RomanNum
Pick an integer from 1 to 1024 to convert to a roman numeral: 1025

Please input a range from 1 to 1024

stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q1_RomanNum
Pick an integer from 1 to 1024 to convert to a roman numeral: 214

Roman numeral is: CCXIV
```

As shown in Figure 1.1, the algorithm returns correct values for the inputted decimal numbers. The algorithm used is a greedy algorithm that always returns the correct solution.

Question 2: Reverse Polish Number

2.1: Algorithm description

The question requires an algorithm that evaluates arithmetic expressions by using reverse polish notation and by using a stack data structure. The required operations are +, -, x, and /.

2.2 Implementation

The program is implemented using a while loop that evaluates to true indefinitely. The program can be terminated if the user inputs the 'q' character.

With each iteration the program reads user input and pushes that input to the stack data structure if that input is a numerical value. Otherwise, if the input is not a numerical value, the program determines whether the input is an operator, a quit command, or a print command and then takes the appropriate actions.

If the input is not a valid numerical value nor a valid command then the program prints to the screen that the input is not valid.

In order to perform some of these actions the program uses several functions which will be discussed in the next section.

2.3: Functions

2.3.1: is_num() function

This function checks whether the inputted text is numeric. It does this by first converting the function argument into a C string by using the .c_str() method. Then, the function makes a call to the strtod() function to convert the string into a double. A pointer to char, endPointer, was initialized to NULL to pass as an argument to strtod().

If the c string is equal to NULL then the function returns false. False is also returned if endPointer is changed to anything that is not equal to NULL or if it points to the beginning of c_input_string. If these two checks do not trigger a false return, a true is returned.

2.3.2: conv_to_double() function

Converts from string to C string by using strod() function.

2.3.3: is_operat()

Checks whether an inputted string is an operator by comparing to a string array which contains the 4 operators. If the operator is found in this array the function returns true. If not found in the array the function returns false.

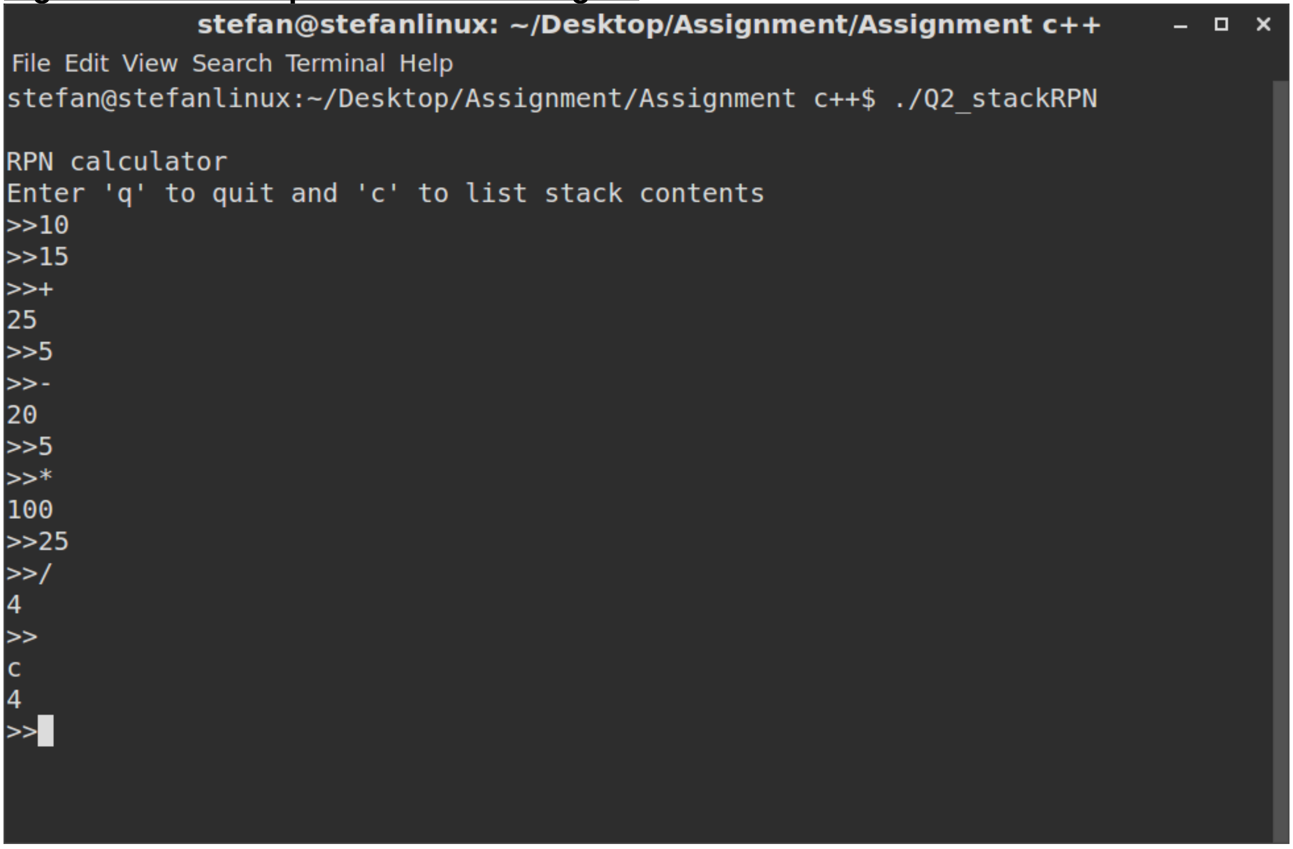
2.3.4: operate()

This function is called when an inputted text is recognized as an operator. Two values are popped off the stack and assigned their respective variables. The appropriate operation is selected depended on the operator. Once the operation is complete the result is pushed back into the stack and printed to the screen.

2.4: Testing

The program is initially tested for basic operations.

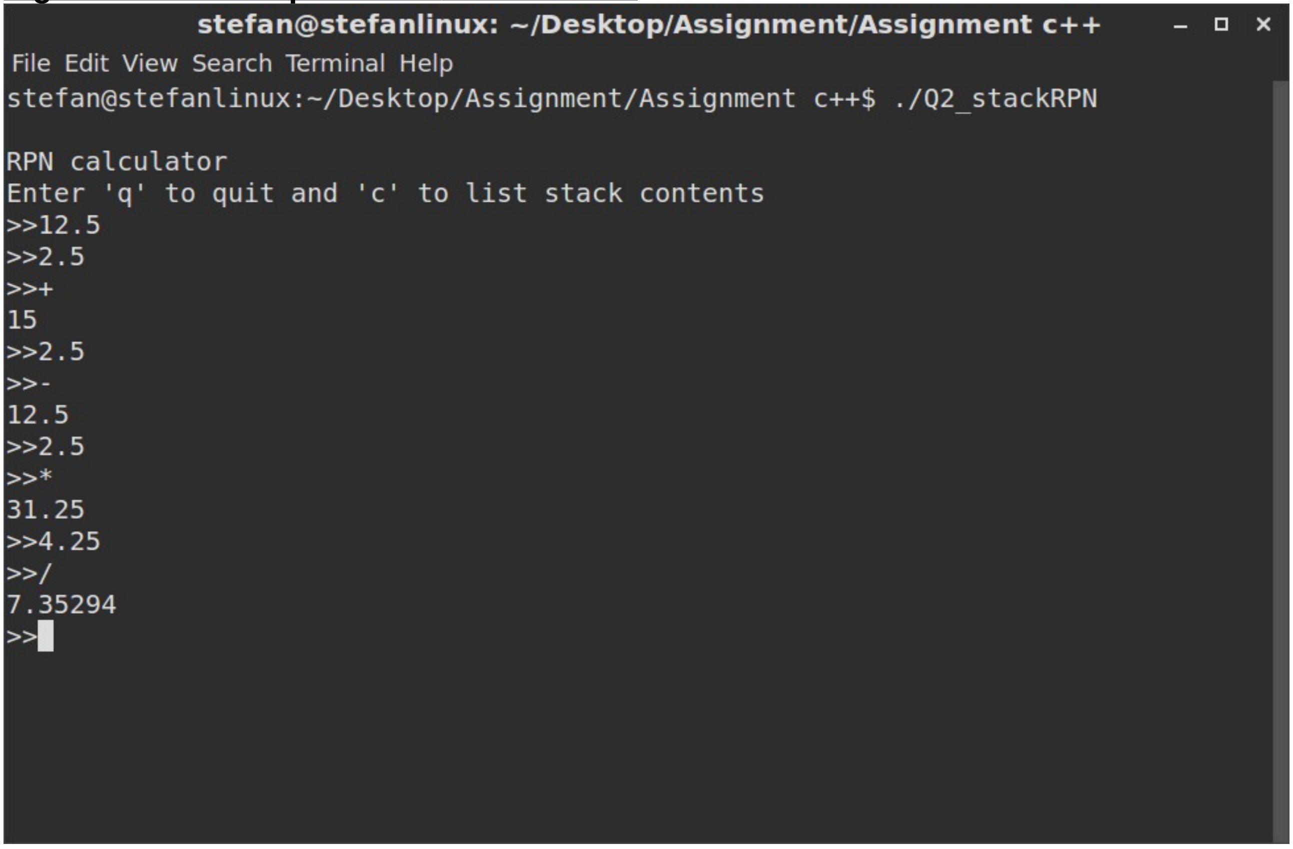
Figure 2.1: Basic operations with integers


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q2_stackRPN

RPN calculator
Enter 'q' to quit and 'c' to list stack contents
>>10
>>15
>>+
25
>>5
>>-
20
>>5
>>*
100
>>25
>>/
4
>>
c
4
>>
```

All operations are correct for integers.

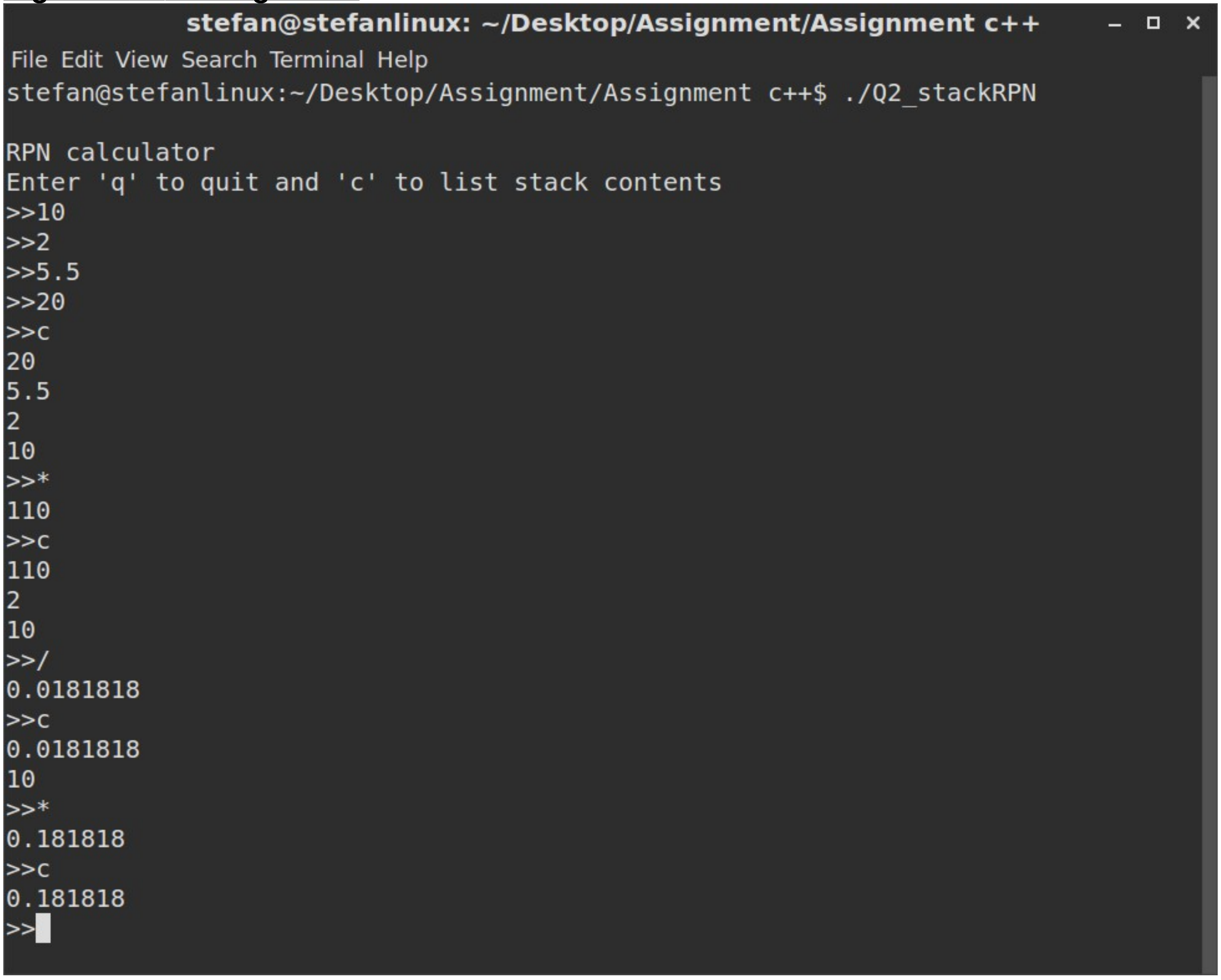
Figure 2.2: Basic operations with doubles


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q2_stackRPN

RPN calculator
Enter 'q' to quit and 'c' to list stack contents
>>12.5
>>2.5
>>+
15
>>2.5
>>-
12.5
>>2.5
>>*
31.25
>>4.25
>>/
7.35294
>>
```

All operations are correct for double operations.

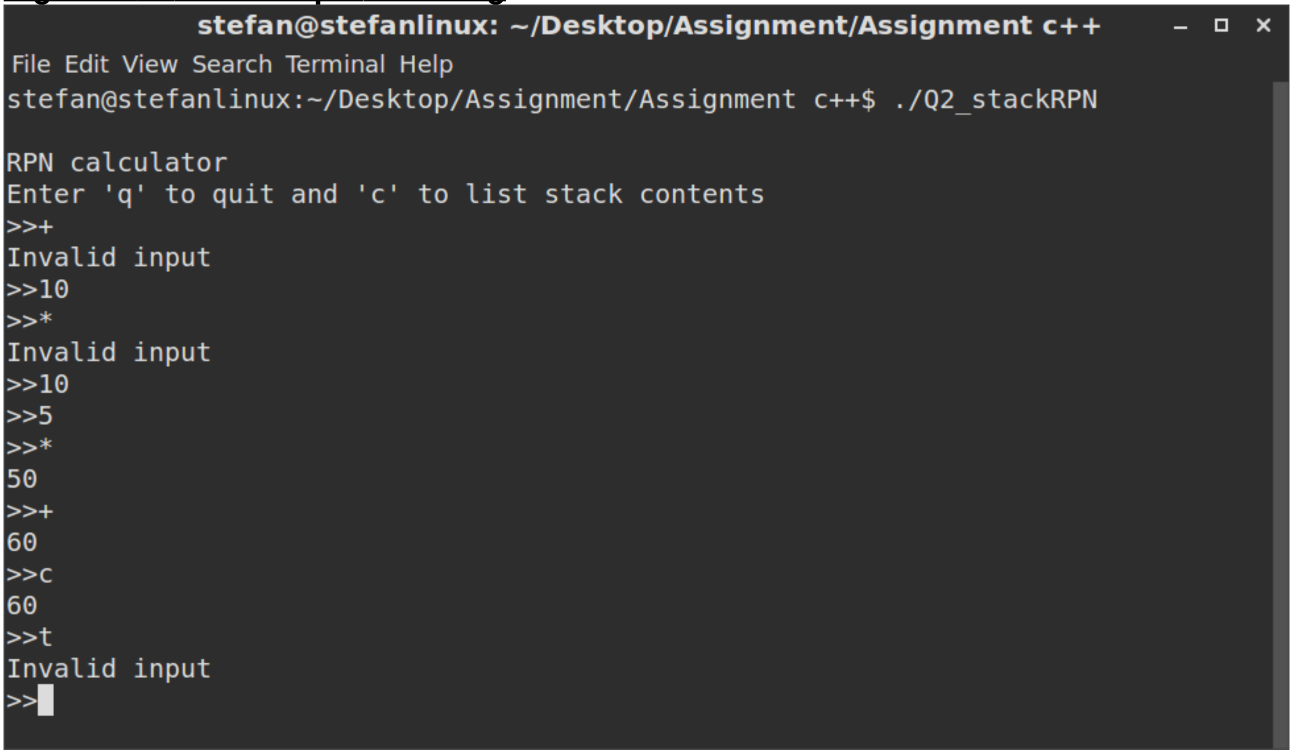
Figure 2.2: Testing Stack


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q2_stackRPN

RPN calculator
Enter 'q' to quit and 'c' to list stack contents
>>10
>>2
>>5.5
>>20
>>c
20
5.5
2
10
>>*
110
>>c
110
2
10
>>/
0.0181818
>>c
0.0181818
10
>>*
0.181818
>>c
0.181818
>>
```

The top value of the stack is set to the right hand side value and the value below is set to the left hand side value. Each operation pops two values from the stack and pushes back a result.

Figure 2.2: Invalid input handling


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q2_stackRPN

RPN calculator
Enter 'q' to quit and 'c' to list stack contents
>>+
Invalid input
>>10
>>*
Invalid input
>>10
>>5
>>*
50
>>+
60
>>c
60
>>t
Invalid input
>>
```

Using the is_num() and is_operat() function the program can determine whether the input is valid. The program also checks whether the stack size is greater than one before performing and operation. If the input is not a number and not an operation, the program checks whether the input is 'q' or 'c'. If none of these checks succeed the program prints “Invalid input” to the screen and waits for the next input.

Question 3: Sieve of Eratosthenes algorithm

3.1: Algorithm description

The algorithm can be implemented by following these 4 steps:

1) Create a list of integers from 2 to n.
2) Let p equal 2
3) Enumerate p's multiples, excluding p, while the multiple of p is smaller or equal to n.
4) The first number greater than p in the enumerated list that is not marked should become the new p. The process is then repeated from step 3. If no number greater than p is found then the process is terminated and all numbers that are not enumerated are prime numbers.

However, modifications are made to the above algorithm to improve performance as discussed in section 3.2: Optimizations.

The algorithm is implemented through a function that takes a natural number as input and outputs a boolean array, where the length is equal to the inputted number + 1, as a result. The resulting boolean array is set to true at indexes that are prime numbers.

The consecutive integers from 2 to n are represented by using a boolean array where the indices represent the particular number. Initially all values are set to true.

A for loop is run that changes all values in the boolean array, where the indices are multiples of 2 (excluding 2), to false. These elements are known to have indices that are not prime due to being multiples of 2.

Another for loop is used starting from 3, the initial value for p, and finishing at the square root of the inputted number, n. This for loop does the same operation as before but for multiples of p instead. 3 is used as a starting point because it is known to be a prime number and it is incremented by 2 for each iteration. This is so that the operation is done only for odd numbers.

Once the for loop finishes the function returns the boolean array as a pointer.

3.2: Optimizations

The first optimization is that the numbers marked as not prime start from p^2 as multiples of p less than p^2 have already been marked.

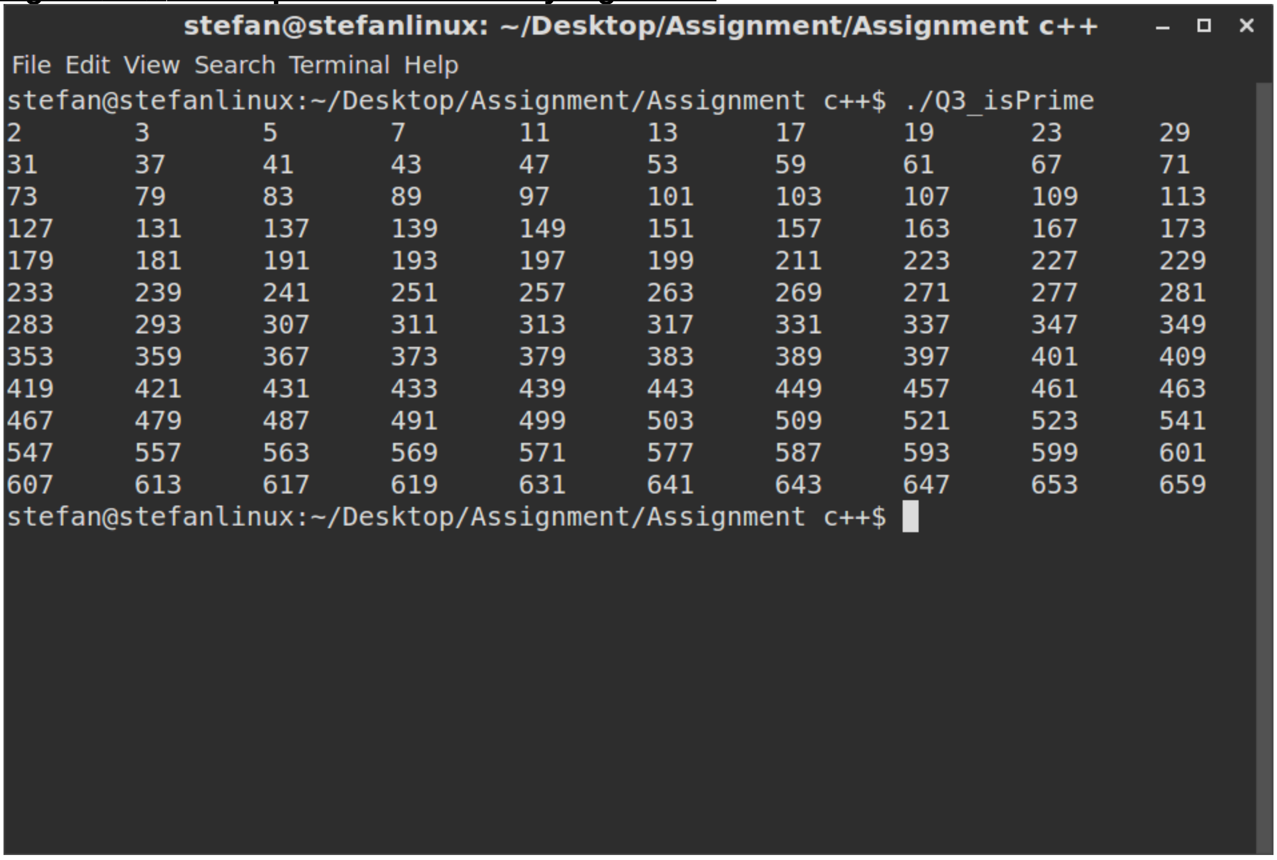
The second optimzation is that the p chosen in the for loop only reaches $n^{0.5}$. This is because the numbers marked as not prime start from p^2. Because of this the algorithm can terminate when p^2 becomes larger than n.

The third optimization is that, excluding p = 2, only odd numbers are chosen for p. This is because primes can only be odd numbers.

3.3: Testing

The main() function tests the algorithm by first calling the function isPrime(). The program then iterates through the boolean array that is returned by isPrime() and prints the index value (which represents the tested number) if that index has a true value in the boolean array. The tested input was 660.

Figure 3.1: List of primes returned by algorithm


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q3_isPrime
2       3       5       7       11      13      17      19      23      29
31      37      41      43      47      53      59      61      67      71
73      79      83      89      97      101     103     107     109     113
127     131     137     139     149     151     157     163     167     173
179     181     191     193     197     199     211     223     227     229
233     239     241     251     257     263     269     271     277     281
283     293     307     311     313     317     331     337     347     349
353     359     367     373     379     383     389     397     401     409
419     421     431     433     439     443     449     457     461     463
467     479     487     491     499     503     509     521     523     541
547     557     563     569     571     577     587     593     599     601
607     613     617     619     631     641     643     647     653     659
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$
```

Question 4: Shell Sort

4.1: Algorithm description

The algorithm starts by comparing elements which are far apart and moving through the array while maintaining the same gap. Once the end of the array is reached the gap size is reduced based on the gap sequence chosen. The gap sequence chosen for this algorithm is that of Ciura (2001) [1]. This is done iteratively until the smallest gap increment is reached.

4.2: Testing

An empty array of size 16384 was filled with random numbers and tested in the main function by using the following code:

```
bool isSorted = true;
for(int i = 1; i < 16384; i++)
{
        if(numArray[i]<numArray[i-1])
        {
                cout << "Sequence not sorted" << endl;
                isSorted = false;
                break;
        }
}
if(isSorted == true)
        cout << "Sequence is sorted" << endl;
```

Each element is compared with the previous element. If any element is smaller than the previous element the iteration is stopped and “Sequence not sorted” is outputted. Otherwise, “Sequence is sorted” outputted.

Question 5: Square root using Newton's method

5.1: Algorithm description

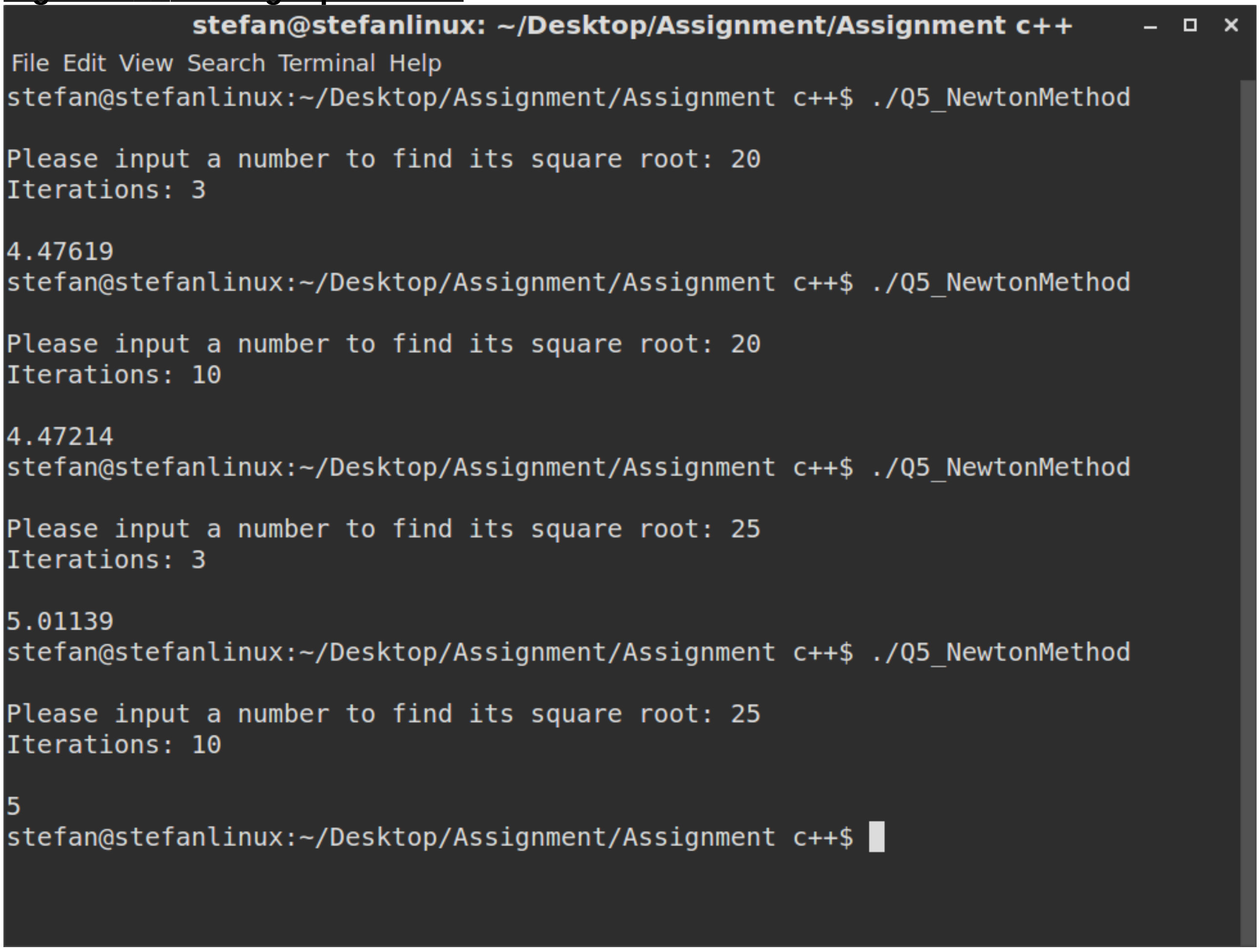
The algorithm consists of a simple program that asks for a number input to be square rooted and another input to set the number of iterations of Newton's method. Iterations are stored as int and the number to be square rooted is stored as a double.

A for loop is run for the set number of iterations where the equation of Newton's method is calculated as shown:

```
for(int index = 0; index < iter; index++)
        result = result - (result*result - numinput)/(2*result);
```

5.2: Testing

Figure 5.1: Finding square root



stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q5_NewtonMethod

Please input a number to find its square root: 20
Iterations: 3

4.47619
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q5_NewtonMethod

Please input a number to find its square root: 20
Iterations: 10

4.47214
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q5_NewtonMethod

Please input a number to find its square root: 25
Iterations: 3

5.01139
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q5_NewtonMethod

Please input a number to find its square root: 25
Iterations: 10

5
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ 
```

The equations works as intended. As expected, increasing the number of iterations increases the accuracy of the algorithm.

Question 6: Multiplying matrices

6.1: Program description

The program creates two matrices filled with random integers with dimensions that user inputs upon prompt. The function used to create these two matrices is randomMatrix().

After the two random matrices are created the program calls the function multMatrices() and stores the result in the result variable.

All matrix variables are of pointer to array of pointers type. Each pointer in the pointer array points to an int array which contains the matrix numbers. The pointer array represents the rows of a matrix while each int array represents the numbers in each row of the same matrix. Writing matrix[x][y] would access the xth row of the yth column from the matrix.

6.2: Functions

6.2.1: printMatrix() function

Takes as input a matrix and its dimensions and prints to screen the contents of the matrix by using two for loops.

6.2.2: randomMatrix() function

Takes as input the dimensions of a desired matrix. Memory is allocated for the matrix rows and columns by using new. Two for loops are used two fill the matrix with random integers. A modulo operator is used such that the values do not exceed an arbitrary limit.

6.2.3: multMatrices() function

Takes as input two matrices, the number of rows and columns of the first matrix, and the columns of the second matrix. Similar to the randomMatrix() function, space is allocated using new.

Multiplication is done by using three for loops. The first iterator goes through the rows of matrix1, the second iterator goes through the columns of matrix2, and the third iterator goes through the columns of matrix1.

The first and second iterator are required to change the element of the result matrix. The third operator is used to sum up the multiplications made for each element of the third matrix.

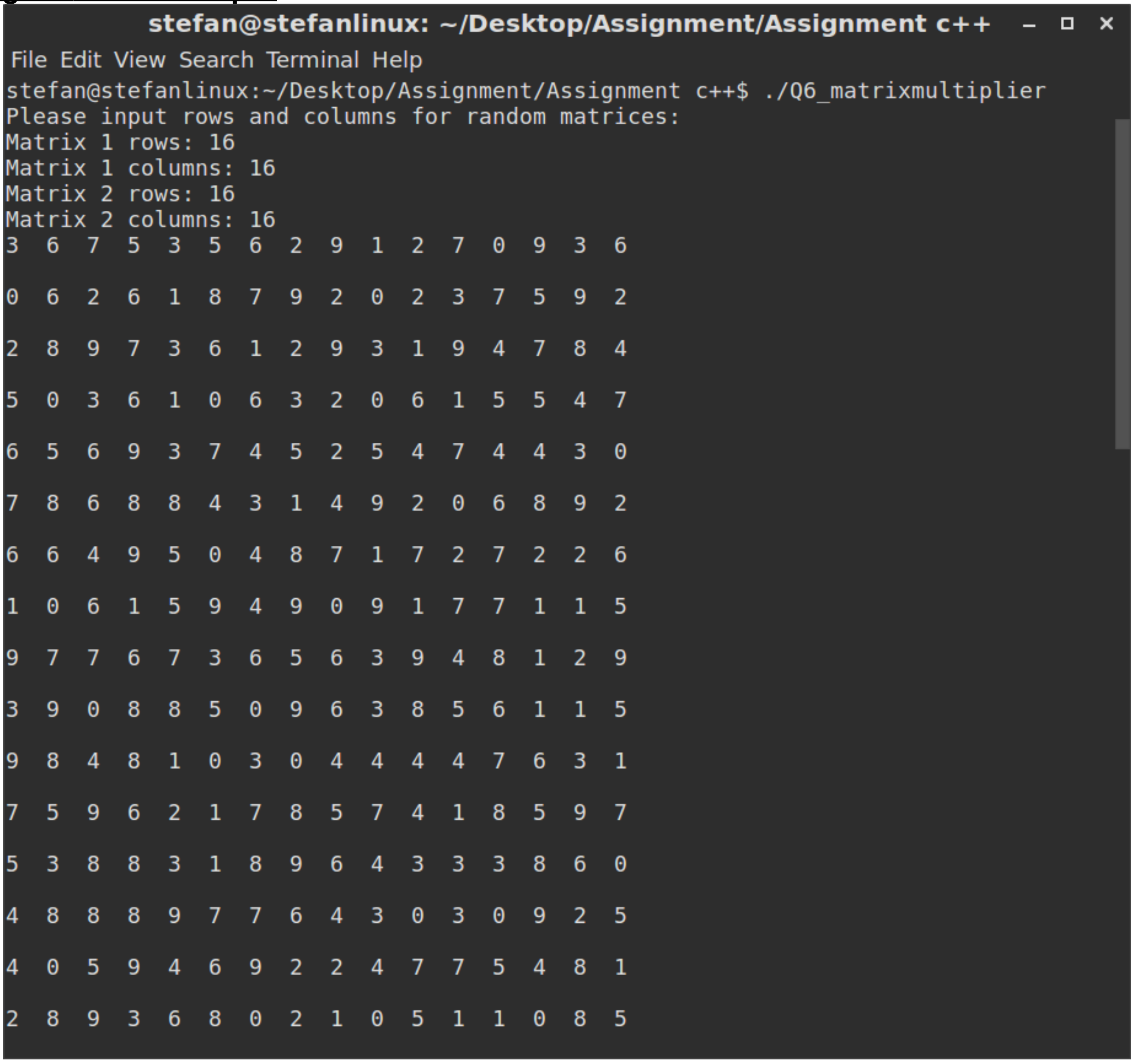
6.2.4: freeMem() function

Takes a matrix as input and frees the memory allocated for that matrix. First each row of the matrix is freed and then the matrix itself is freed. Since the c++ construct for memory allocation, new, was used, this function uses delete to free memory.

6.3: Testing

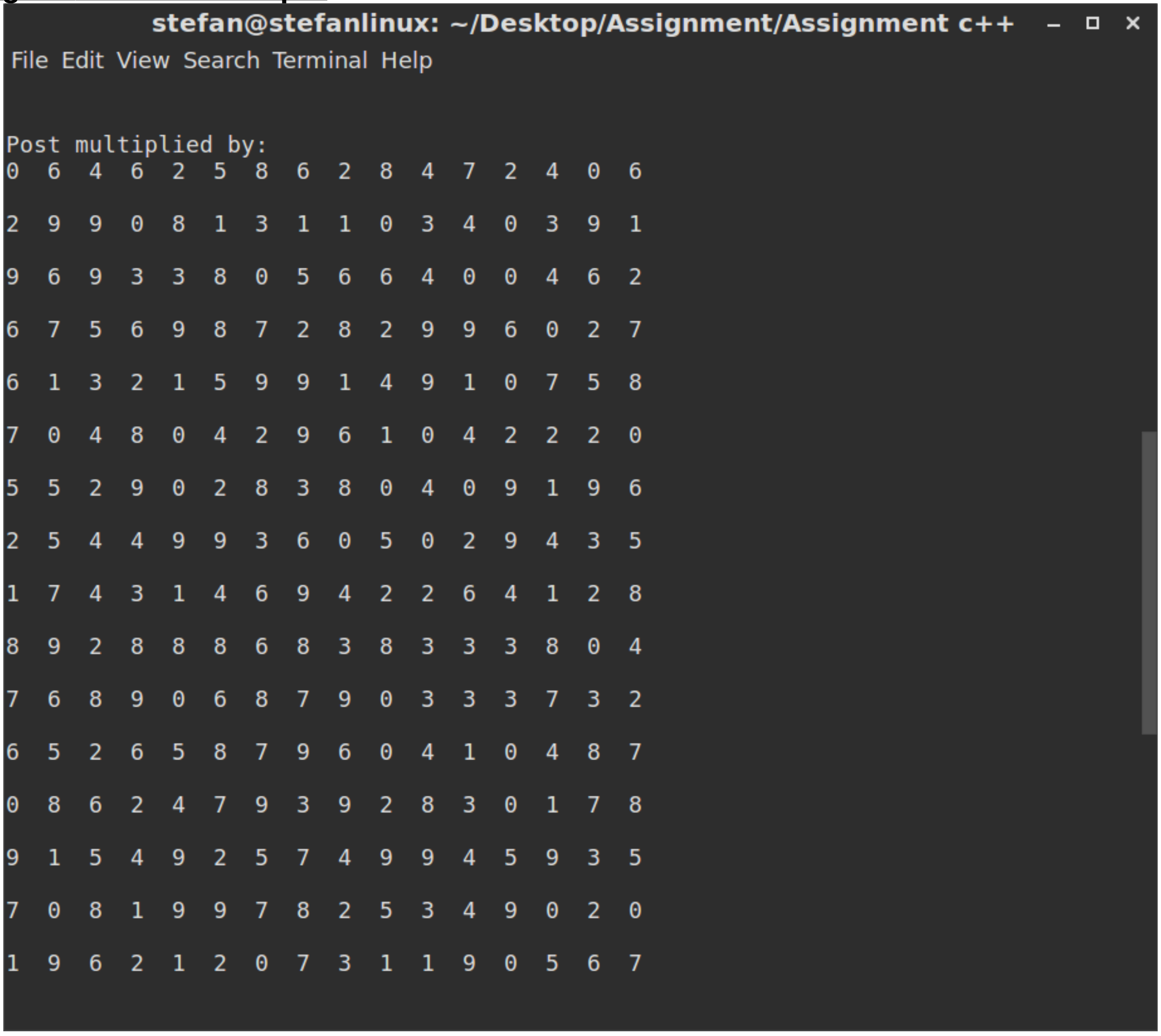
Testing is done by running the program and inputting the appropriate dimensions. In this case the dimensions were 16x16.

Figure 6.1: First input


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

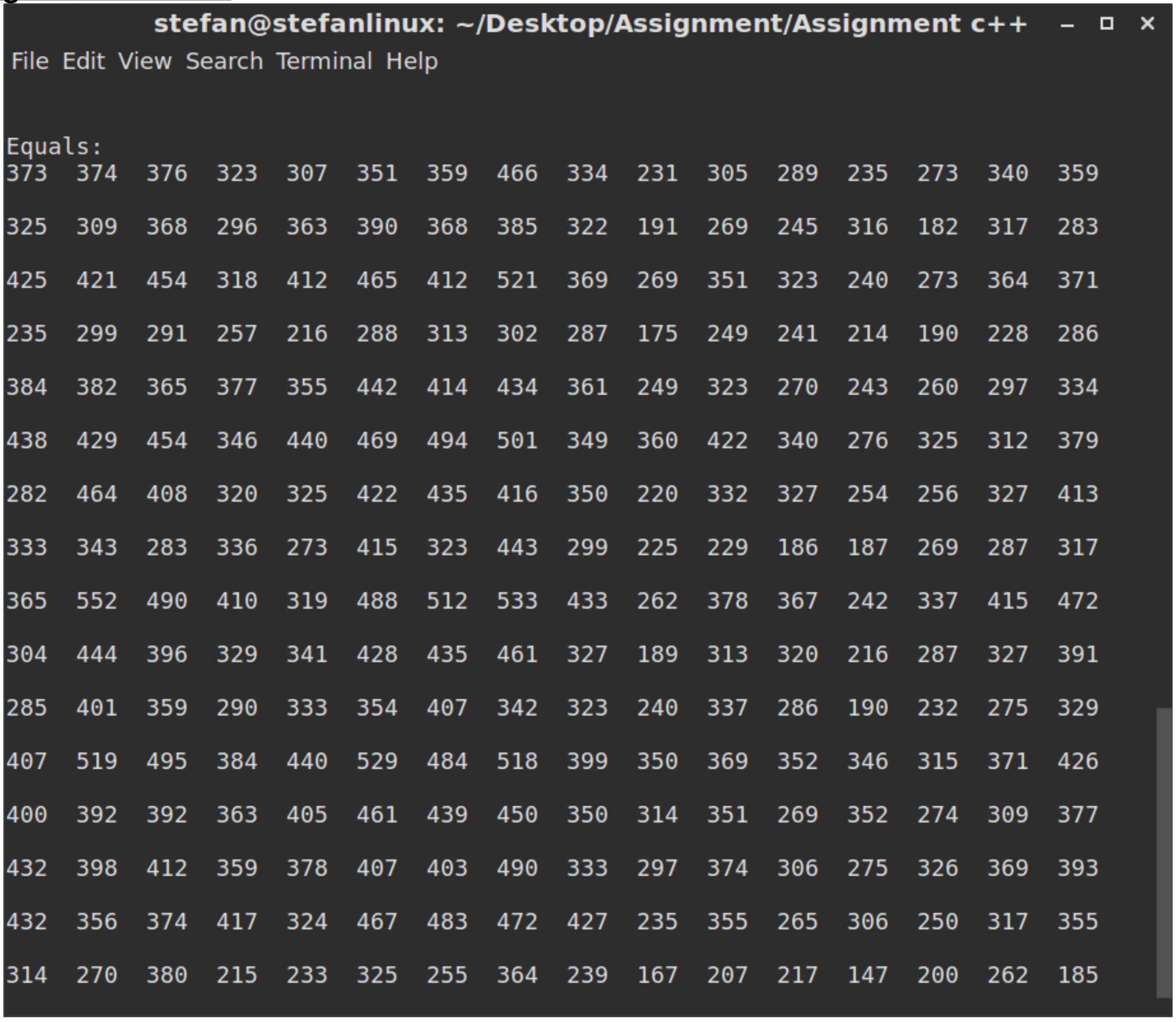
```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q6_matrixmultiplier
Please input rows and columns for random matrices:
Matrix 1 rows: 16
Matrix 1 columns: 16
Matrix 2 rows: 16
Matrix 2 columns: 16
3  6  7  5  3  5  6  2  9  1  2  7  0  9  3  6

0  6  2  6  1  8  7  9  2  0  2  3  7  5  9  2

2  8  9  7  3  6  1  2  9  3  1  9  4  7  8  4

5  0  3  6  1  0  6  3  2  0  6  1  5  5  4  7

6  5  6  9  3  7  4  5  2  5  4  7  4  4  3  0

7  8  6  8  8  4  3  1  4  9  2  0  6  8  9  2

6  6  4  9  5  0  4  8  7  1  7  2  7  2  2  6

1  0  6  1  5  9  4  9  0  9  1  7  7  1  1  5

9  7  7  6  7  3  6  5  6  3  9  4  8  1  2  9

3  9  0  8  8  5  0  9  6  3  8  5  6  1  1  5

9  8  4  8  1  0  3  0  4  4  4  4  7  6  3  1

7  5  9  6  2  1  7  8  5  7  4  1  8  5  9  7

5  3  8  8  3  1  8  9  6  4  3  3  3  8  6  0

4  8  8  8  9  7  7  6  4  3  0  3  0  9  2  5

4  0  5  9  4  6  9  2  2  4  7  7  5  4  8  1

2  8  9  3  6  8  0  2  1  0  5  1  1  0  8  5
```

Figure 6.2: Second input


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
Post multiplied by:
0  6  4  6  2  5  8  6  2  8  4  7  2  4  0  6

2  9  9  0  8  1  3  1  1  0  3  4  0  3  9  1

9  6  9  3  3  8  0  5  6  6  4  0  0  4  6  2

6  7  5  6  9  8  7  2  8  2  9  9  6  0  2  7

6  1  3  2  1  5  9  9  1  4  9  1  0  7  5  8

7  0  4  8  0  4  2  9  6  1  0  4  2  2  2  0

5  5  2  9  0  2  8  3  8  0  4  0  9  1  9  6

2  5  4  4  9  9  3  6  0  5  0  2  9  4  3  5

1  7  4  3  1  4  6  9  4  2  2  6  4  1  2  8

8  9  2  8  8  8  6  8  3  8  3  3  3  8  0  4

7  6  8  9  0  6  8  7  9  0  3  3  3  7  3  2

6  5  2  6  5  8  7  9  6  0  4  1  0  4  8  7

0  8  6  2  4  7  9  3  9  2  8  3  0  1  7  8

9  1  5  4  9  2  5  7  4  9  9  4  5  9  3  5

7  0  8  1  9  9  7  8  2  5  3  4  9  0  2  0

1  9  6  2  1  2  0  7  3  1  1  9  0  5  6  7
```

Figure 6.3: Result



stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++

File Edit View Search Terminal Help

```
Equals:
373  374  376  323  307  351  359  466  334  231  305  289  235  273  340  359

325  309  368  296  363  390  368  385  322  191  269  245  316  182  317  283

425  421  454  318  412  465  412  521  369  269  351  323  240  273  364  371

235  299  291  257  216  288  313  302  287  175  249  241  214  190  228  286

384  382  365  377  355  442  414  434  361  249  323  270  243  260  297  334

438  429  454  346  440  469  494  501  349  360  422  340  276  325  312  379

282  464  408  320  325  422  435  416  350  220  332  327  254  256  327  413

333  343  283  336  273  415  323  443  299  225  229  186  187  269  287  317

365  552  490  410  319  488  512  533  433  262  378  367  242  337  415  472

304  444  396  329  341  428  435  461  327  189  313  320  216  287  327  391

285  401  359  290  333  354  407  342  323  240  337  286  190  232  275  329

407  519  495  384  440  529  484  518  399  350  369  352  346  315  371  426

400  392  392  363  405  461  439  450  350  314  351  269  352  274  309  377

432  398  412  359  378  407  403  490  333  297  374  306  275  326  369  393

432  356  374  417  324  467  483  472  427  235  355  265  306  250  317  355

314  270  380  215  233  325  255  364  239  167  207  217  147  200  262  185
```

The resulting matrix is correct.

Question 7: Largest number in list using recursion

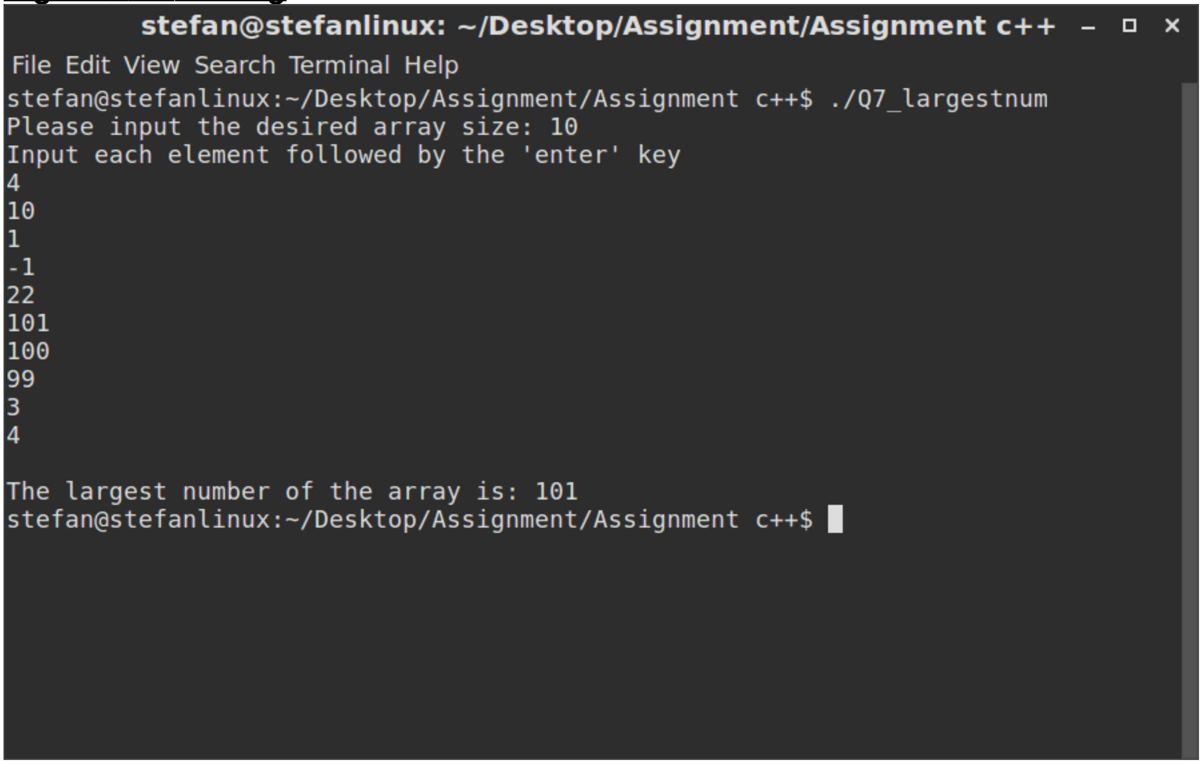
7.1: Algorithm description

The algorithm is written in a recursive function that takes the array as input and an integer size. In the base case, that is when the size of the array is 1, the funtion simply returns the array. In the general case, the function either returns the last element of the array or makes a recursive call to find the largest number of the remaining elements depending on which is largest.

7.2: Testing

Testing is done by using a program that uses the findLargest() function. First it prompts the user for an array size. The program then asks the user to input the numbers into the array for testing purposes. The inputted values are pushed into a c++ vector and then converted into an array. The findLargest() function is then called.

Figure 7.1: Testing


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q7_largestnum
Please input the desired array size: 10
Input each element followed by the 'enter' key
4
10
1
-1
22
101
100
99
3
4

The largest number of the array is: 101
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ 
```

Question 8: Series expansion

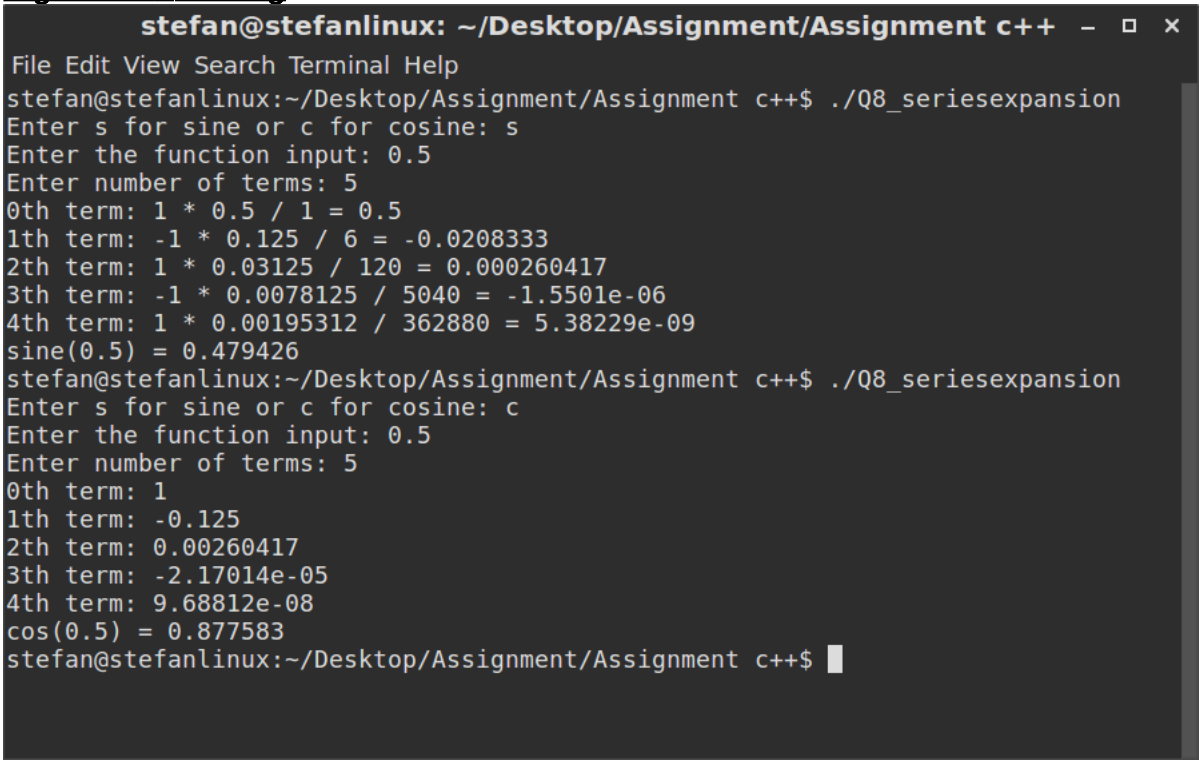
8.1: Algorithm description

The algorithm is simply a for loop that calculates the expressions for the expansions of either sine or cosine. Each iteration adds to the precision of the returned result.

8.2: Testing

Testing is done through a program that requests user input. User input determines whether a sine or cosine calculation is performed, the value being calculated, and the number of terms, or itereations, the function is to perform. If the the number being evaluated is relatively large, say 10, then the number of iterations must be increased for a correct answer.

Figure 8.1: Testing


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q8_seriesexpansion
Enter s for sine or c for cosine: s
Enter the function input: 0.5
Enter number of terms: 5
0th term: 1 * 0.5 / 1 = 0.5
1th term: -1 * 0.125 / 6 = -0.0208333
2th term: 1 * 0.03125 / 120 = 0.000260417
3th term: -1 * 0.0078125 / 5040 = -1.5501e-06
4th term: 1 * 0.00195312 / 362880 = 5.38229e-09
sine(0.5) = 0.479426
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q8_seriesexpansion
Enter s for sine or c for cosine: c
Enter the function input: 0.5
Enter number of terms: 5
0th term: 1
1th term: -0.125
2th term: 0.00260417
3th term: -2.17014e-05
4th term: 9.68812e-08
cos(0.5) = 0.877583
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$
```

Question 9: Palindrome

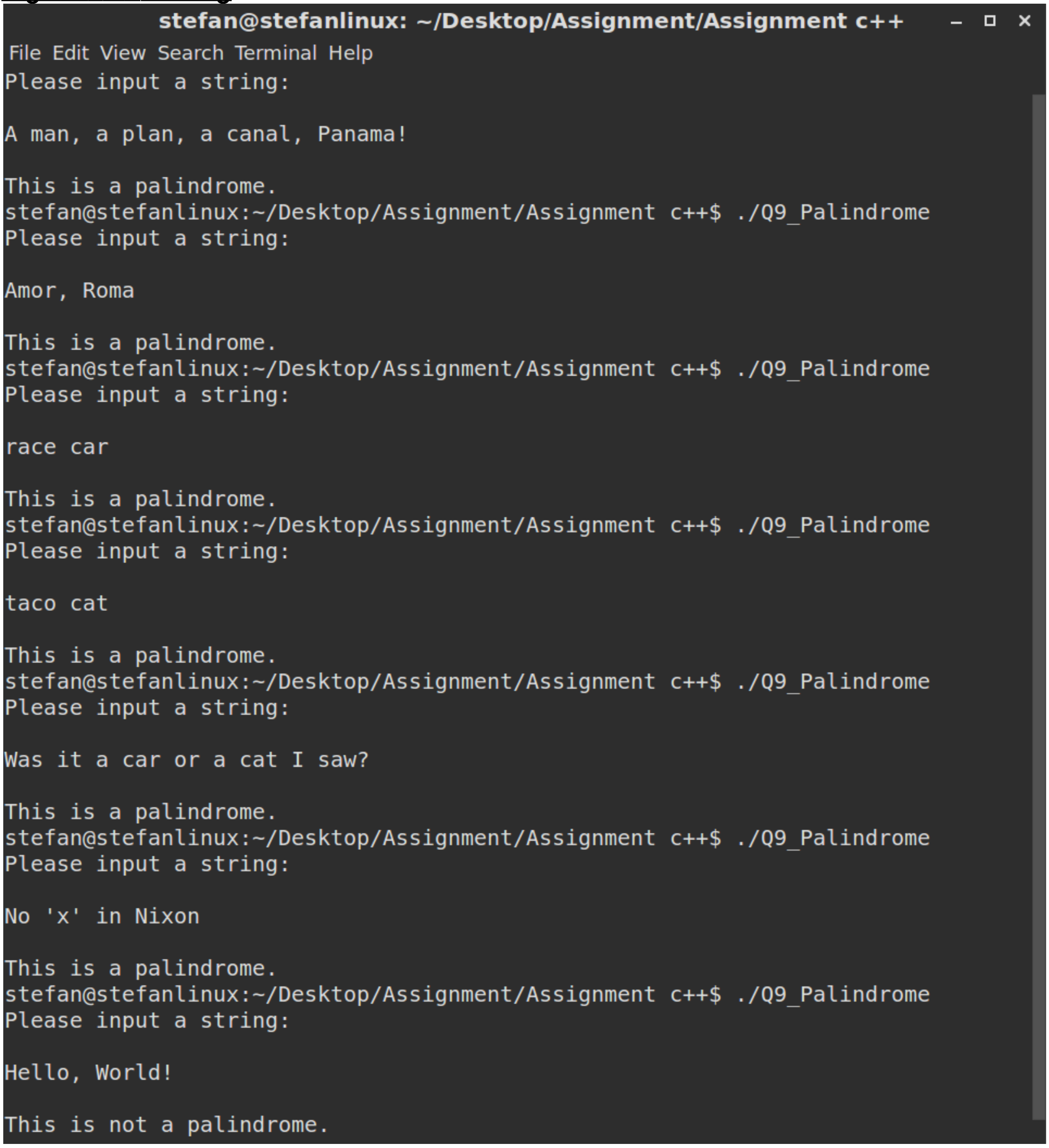
9.1: Algorithm description

Before checking whether the string is a palindrome the algorithm iterates through the string and selects only alphabetical characters. Uppercase letters are converted to lower case letters. Selected letters are appended to a vector. The algorithm then performs a for loop to compare elements from the vector. On the first iteration, the first and last elements are compared. On successive iterations the algorithm moves towards the middle of the vector until no elements remain or one element remains. If all comparisons turn out to be equal the function returns true. Otherwise, the functions returns false.

9.2: Testing

The function is tested by prompting for a string and passing to the isPalindrome function.

Figure 9.1: Testing


stefan@stefanlinux: ~/Desktop/Assignment/Assignment c++
File Edit View Search Terminal Help

```
Please input a string:

A man, a plan, a canal, Panama!

This is a palindrome.
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q9_Palindrome
Please input a string:

Amor, Roma

This is a palindrome.
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q9_Palindrome
Please input a string:

race car

This is a palindrome.
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q9_Palindrome
Please input a string:

taco cat

This is a palindrome.
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q9_Palindrome
Please input a string:

Was it a car or a cat I saw?

This is a palindrome.
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q9_Palindrome
Please input a string:

No 'x' in Nixon

This is a palindrome.
stefan@stefanlinux:~/Desktop/Assignment/Assignment c++$ ./Q9_Palindrome
Please input a string:

Hello, World!

This is not a palindrome.
```

As shown, spaces and punctuation are ignored by the algorithm.

Appendix

Question 1

```
#include <iostream>
#include <string>

using namespace std;
string dec_to_rom(unsigned int value);

int main()
{
	int numinput;
	cout << "Pick an integer from 1 to 1024 to convert to a roman numeral: ";
	cin >> numinput;

	if(numinput < 1 || numinput > 1024)
		cout << "\nPlease input a range from 1 to 1024\n" << endl;
	else
		cout << "\nRoman numeral is: " + dec_to_rom(numinput) + '\n' << endl;

	return 0;

}

string dec_to_rom(unsigned int value)
{
   struct Rom_node
	{
		unsigned int Dec_num;
		char const* Rom_num;
	};



   const struct Rom_node roman_array[] =
   {
        {1000, "M"}, {900, "CM"},
        {500, "D"}, {400, "CD"},
        {100, "C"}, { 90, "XC"},
        { 50, "L"}, { 40, "XL"},
        { 10, "X"}, { 9, "IX"},
        { 5, "V"}, { 4, "IV"},
        { 1, "I"},
        { 0, NULL}
    };

        string result;
        int roman_index = 0;

        while(value > 0)
        {
            if(value >= roman_array[roman_index].Dec_num)
            {
                result += roman_array[roman_index].Rom_num;
                value -= roman_array[roman_index].Dec_num;
            }
            else
                roman_index += 1;
        }

        return result;
}
```

Question 2

```
#include <iostream>
#include <stack>
#include <string>
#include <stdlib.h>

using namespace std;

bool is_num(const string& input_string);
double conv_to_double(const string& input_string);
bool is_operat(const string& input_string);
void operate(const string& input_string, stack<double>& rpnStack);
void printcontents(stack<double>& rpnStack);

int main()
{
	stack<double> rpnStack;

	cout << "\nRPN calculator\nEnter 'q' to quit and 'c' to list stack contents" << endl;
	string input_string;

	while(true)
	{
		cout << ">>";

		cin >> input_string;

		double num;
		if(is_num(input_string))
			rpnStack.push(conv_to_double(input_string));

		else if(is_operat(input_string) && rpnStack.size()>1)
			operate(input_string, rpnStack);

		else if(input_string == "q")
```

```
				return 0;

			else if(input_string == "c")
				printcontents(rpnStack);

			else
				cout << "Invalid input" << endl;



		}

		return 0;
	}

	bool is_num(const string& input_string)
	{
		const char * c_input_string = input_string.c_str();
		//convert to string

		char* endPointer = '\0';
		strtod(c_input_string, &endPointer);

		if(*c_input_string == '\0')
			return false;

		if(*endPointer !=  '\0' || endPointer == c_input_string)
			return false;

		return true;
	}

	double conv_to_double(const string& input_string)
	{
		const char * c_input_string = input_string.c_str();
		char* endPointer = '\0';

		return strtod(c_input_string, &endPointer);
```

```
}



bool is_operat(const string& input_string)
{
	string operators[] = {"+", "-", "*", "/"};

	for(int iter = 0; iter < 4; iter++)
	{
		if(input_string == operators[iter])
			return true;
	}
	return false;
}

void operate(const string& input_string, stack<double>& rpnStack)
{
	double lvalue, rvalue, result;

	rvalue = rpnStack.top();
	rpnStack.pop();

	lvalue = rpnStack.top();
	rpnStack.pop();

	if(input_string == "+")
		result = lvalue + rvalue;

	else if(input_string == "-")
		result = lvalue - rvalue;

	else if(input_string == "*")
		result = lvalue * rvalue;

	else if(input_string == "/")
		result = lvalue / rvalue;
```

```
        cout << result << endl;
        rpnStack.push(result);



}

void printcontents(stack<double>& rpnStack)
{
        for(stack<double> copy = rpnStack; !copy.empty(); copy.pop())
                cout << copy.top() << endl;

}
```

Question 3

```
#include <iostream>
#include <math.h>

using namespace std;

bool * isPrime(int input_num);

int main()
{
	int primes_up_to;
	cout << "Enter a number up to which all primes will be listed:";
	cin >> primes_up_to;
	bool * primelist = isPrime(primes_up_to);

	int newline_count = 0;
	for(int index = 2; index < primes_up_to; index++)
	{
		if(newline_count == 10)
			{
				cout << endl;
				newline_count = 0;
			}
		if(primelist[index] == 1)
		{
			cout << index << '\t';
			newline_count++;
		}
	}
	delete primelist;



	cout << endl;
return 0;


}


bool * isPrime(int input_num)
{
	bool * natural_numbers = new bool[input_num + 1];
	fill_n(natural_numbers, input_num + 1, true);  //a boolean array for marking numbers from 2 to n where any index i equals the natural number i+2 (index 0 equals 2, index n-2 equals n)
	int prime;
	int multiplier;


	int p = 2;
	for(p*p + p*multiplier; p*p + p*multiplier <= input_num; multiplier++)
		natural_numbers[p*p + p*multiplier] = false;



	for(p=3; p<=pow(input_num, 0.5); p += 2) //not exceeding sqrt n
	{


		multiplier = 0;
		for(p*p + p*multiplier; p*p + p*multiplier <= input_num; multiplier++)
			natural_numbers[p*p + p*multiplier] = false;


	}
	return natural_numbers;
}
```

Question 4

```
#include <iostream>
#include <cstdlib>

using namespace std;

int * shellSort(int * numArray);

int main()
{
	int numArray[16384];

	for(int i = 0; i < 16384; i++)
		numArray[i] = rand();

/* Uncomment to print unsorted array
	for(int i = 0; i < 16384; i++)
	{
		cout << i << ": " << numArray[i] << '\t';

		if(i%5 == 0)
			cout << endl;
	}
	cout << endl;

*/

	shellSort(numArray);

/* Uncomment to print sorted array
	for(int i = 0; i < 16384; i++)
	{
		cout << i << ": " << numArray[i] << '\t';

		if(i%5 == 0)
```

```
				cout << endl;
		}
		cout << endl;

*/
		bool isSorted = true;
		for(int i = 1; i < 16384; i++)
		{
			if(numArray[i]<numArray[i-1])
			{
				cout << "Sequence not sorted" << endl;
				isSorted = false;
				break;
			}
		}
		if(isSorted == true)
			cout << "Sequence is sorted" << endl;

}



int* shellSort(int * numArray)
{
		//gap sequence with 8 elems
		int gapsequence[] = {701, 301, 132, 57, 23, 10, 4, 1};
		int index;
		int index1;
		int index2;
		int index3;

		for(index1 = 0; index1 < 8; index1++)
		{
			for(int index2 = gapsequence[index1]; index2<16384; index2++)
			{
				int temp = numArray[index2];

				for(index3 = index2; index3 >= gapsequence[index1] && numArray[index3-
gapsequence[index1]] > temp; index3 -= gapsequence[index1])
                        {
                                numArray[index3] = numArray[index3-gapsequence[index1]];

                        }

                        numArray[index3] = temp;
                }

        }
}
```

Question 5

```
#include <iostream>

using namespace std;

int main()
{
        double numinput;
        double result =2;
        int iter;

        cout << "\nPlease input a number to find its square root: ";
        cin >> numinput;
        cout << "Iterations: ";
        cin >> iter;
        cout << endl;



        for(int index = 0; index < iter; index++)
                result = result - (result*result - numinput)/(2*result);

        cout << result << endl;

}
```

Question 6

```
#include <iostream>
#include <cstdlib>

using namespace std;

void printMatrix(int ** Matrix, int columns, int rows);
int** randomMatrix(int columns, int rows);
int** multMatrices(int ** matrix1, int ** matrix2, int rowsMatrix1, int colsMatrix2, int
colsMatrix1);
void freeMem(int ** Matrix, int columns);

int main()
{

        int rowsMatrix1;
        int colsMatrix1;
        int rowsMatrix2;
        int colsMatrix2;

        cout << "Please input rows and columns for random matrices:" << endl;

        cout << "Matrix 1 rows: ";
        cin >> rowsMatrix1;

        cout << "Matrix 1 columns: ";
        cin >> colsMatrix1;

        cout << "Matrix 2 rows: ";
        cin >> rowsMatrix2;

        cout << "Matrix 2 columns: ";
        cin >> colsMatrix2;

        if(colsMatrix1 != rowsMatrix2)
        {
                cout << "Matrix 1 columns must match Matrix 2 rows" << endl;
                return 0;
        }

        int ** matrix1 = randomMatrix(colsMatrix1, rowsMatrix1);
        int ** matrix2 = randomMatrix(colsMatrix2, rowsMatrix2);

        int ** result = multMatrices(matrix1, matrix2, rowsMatrix1, colsMatrix2, colsMatrix1);



        printMatrix(matrix1, colsMatrix1, rowsMatrix1);
        cout << endl << "Post multiplied by:" << endl;
        printMatrix(matrix2, colsMatrix2, rowsMatrix2);
        cout << endl << "Equals:" << endl;
        printMatrix(result, colsMatrix2, rowsMatrix1);
        cout << endl;

        freeMem(result, rowsMatrix1);
        freeMem(matrix1, rowsMatrix1);
        freeMem(matrix2, rowsMatrix2);

        return 0;
}

void printMatrix(int ** Matrix, int columns, int rows)
{
        for(int i = 0; i < rows; i++)
        {
                for(int j = 0; j < columns; j++)
                        cout << Matrix[i][j] << "  ";

                cout << endl << endl;
        }
}

int ** randomMatrix(int columns, int rows)
```

```
{

	int ** matrix;
	int index1, index2;

	matrix = new int *[rows];

	for(int i = 0; i < rows; i++)
		matrix[i] = new int[columns];




	for(index1 = 0; index1 < rows; index1++)
	{
		for(index2 = 0; index2 < columns; index2++)
			matrix[index1][index2] = rand() % 10;
	}

	return (int**) matrix;
}

int** multMatrices(int ** matrix1, int ** matrix2, int rowsMatrix1, int colsMatrix2, int
colsMatrix1)
{
	int** result;
	result = new int *[rowsMatrix1];
	for(int i = 0; i < rowsMatrix1; i++)
		result[i] = new int [colsMatrix2];

	for(int i = 0; i < rowsMatrix1; i++)
	{
		for(int j = 0; j < colsMatrix2; j++)
		{
			result[i][j] = 0;

			for(int x = 0; x < colsMatrix1; x++)
				result[i][j] += matrix1[i][x]*matrix2[x][j];
		}
	}
	return (int**) result;
}


void freeMem(int ** Matrix, int rows)
{
	for(int i = 0; i < rows; i++)
		delete [] Matrix[i];

	delete [] Matrix;
}
```

Question 7

```
#include <iostream>
#include <cstdlib>
#include <vector>


using namespace std;

int findLargest(int numArray[], int size);
int max(int num1, int num2);

int main()
{

    vector<int> numVect;
    int arraySize;
    cout << "Please input the desired array size: ";
    cin >> arraySize;

    int tempInt;

    cout << "Input each element followed by the 'enter' key" << endl;
    for(int i = 0; i < arraySize; i++)
    {
        cin >> tempInt;
        numVect.push_back(tempInt);
    }



    int numArray[arraySize];
    copy(numVect.begin(), numVect.end(), numArray);

    int size = sizeof(numArray)/sizeof(numArray[0]);
```

```
	cout << endl << "The largest number of the array is: " << findLargest(numArray, size) <<
endl;


	return 0;
}

int findLargest(int numArray[], int size)
{
	if(size == 1)
		return numArray[0];


	else
		return max(numArray[size-1], findLargest(numArray, size-1));
}

int max(int num1, int num2)
{
	if(num1 > num2)
		return num1;
	else
		return num2;
}
```

Question 8

```
#include <iostream>
#include <math.h>

using namespace std;

double sineOrCosine(double input, int terms, bool sine);
long double factorial(int n);



int main()
{
	double input;
	char charInput;
	bool sine;
	int terms;

	cout << "Enter s for sine or c for cosine: ";
	cin >> charInput;



	cout << "Enter the function input: ";
	cin >> input;

	cout << "Enter number of terms: ";
	cin >> terms;



	if(charInput == 's')
		cout << "sine(" << input << ") = " << sineOrCosine(input, terms, true) << endl;
	else if(charInput == 'c')
		cout << "cos(" << input << ") = " << sineOrCosine(input, terms, false) << endl;

	return 0;

}
```

```
double sineOrCosine(double input, int terms, bool sine)
{

        long double result = 0;
        long double exponent;
        long double constant;


        if(sine == true)
        {
                for(int i = 0; i < terms; i++)
                {
                        exponent = i*2+1;
                        constant = pow(-1,i)/factorial(i*2+1);
                        result += pow(input, exponent) * constant;

                        cout << i << "th term: ";
                        cout << pow(-1,i) << " * " << pow(input, exponent) << " / ";
                        cout << factorial(i*2+1) << " = " << pow(input, exponent) * constant <<
endl;

                }

                return result;
        }


        else
        {
                for(int i = 0; i < terms; i++)
                {
                        exponent = i*2;
                        constant = pow(-1,i)/factorial(i*2);
                        result += pow(input, exponent) * constant;
```

```
                    cout << i << "th term: " << pow(input, exponent) * constant << endl;
                }

                return result;
            }
        }


long double factorial(int number)
{
  long double factorial = 1;

  for (int i = 1; i <= number; i++)
    factorial = factorial * i;

  return factorial;
}
```

Question 9

```
#include <iostream>
#include <vector>
#include <locale>

using namespace std;

bool isPalindrome(string inputstring);

int main()
{
	string input_string;

   cout << "Please input a string:\n" << endl;
	getline(cin, input_string);
	//cin stops at ' ' so getline is used instead
	cout << endl;

	if(isPalindrome(input_string))
		cout << "This is a palindrome." << endl;

	else
		cout << "This is not a palindrome." << endl;

	return 0;

}

bool isPalindrome(string input_string)
{
	vector<char> charVector;
	for(int iter = 0; iter < input_string.length(); iter++)
	{	//check if either uppercase or lowercase letters
		//ignore all other chars
```

```
        if(isupper(input_string[iter]))
            charVector.push_back(tolower(input_string[iter]));

        else if(islower(input_string[iter]))
            charVector.push_back(input_string[iter]);
    }

    int     size = charVector.size();

    for(int iter = 0; iter < size/2; iter++)
    {       //iterate from beginning and end towards middle and compare
        if (charVector[iter] != charVector[(size-1)-iter])
            return false;
    }
    return true;
}
```

References

[1] Ciura, Marcin (2001). "Best Increments for the Average Case of Shellsort". In Freiwalds, Rusins. *Proceedings of the 13th International Symposium on Fundamentals of Computation Theory* (PDF). London: Springer-Verlag. pp. 106–117. ISBN 3-540-42487-3.