

ELP780 Software Lab

Lab-2

Sah Swapnil Agrawal
2018EET2554
2018-20

August 6, 2019

A report for the lab on C programming



Department of Electrical Engineering
IIT DELHI
India

Contents

1	Problem Statement 1	1
1.1	Problem statement	1
1.2	Assumptions	1
1.3	Algorithms and Implementation	1
1.4	Input and Output format	2
1.5	Test cases	3
1.6	Difficulties /Issues faced	3
1.7	Screenshots	3
2	Problem Statement 2	3
2.1	Problem statement	3
2.2	Assumptions	4
2.3	Algorithms and Implementation	4
2.4	Input and Output format	4
2.5	Test cases	4
2.6	Difficulties /Issues faced	5
2.7	Screenshots	5
3	Appendix	8
3.1	Appendix A: Code for ps1	8
3.2	Appendix B: Code for ps2	14
	References	21

1 Problem Statement 1

1.1 Problem statement

Consider a square matrix of size N ($N \times N$ matrix). Each element is of type unsigned integer.

Let r_i denote the minimum value in the i -th row, and c_i denote the maximum value in i -th column. You have to obtain all such r_i s and c_i s. From now on, let's call the set of all such r_i s 'S1', set of all c_i s 'S2' and set containing both r_i s and c_i s as 'S'. $i = 0$ to $N-1$ (20 min) You have to do the following operations:

- From the set 'S', find the prime numbers and display those values on the terminal (10 min)
- Consider string D as a concatenation of all the r_i s and c_i s e.g. If 'S' contains elements 13,10,5,7,10,7, string D will become 131057107.
- Now substring will be entered by the user and the program should be able to find the position of the first occurrence of that substring in D. Display that position along with string D on the terminal. If the substring is not found then display the message stating no substring matched. (30 min)
- Find the local minima and local maxima and display the index at which they occur in the set 'S'. If no minima and maxima are found, display the message accordingly. (30 min)

We have to write a C program to perform some operations on the data obtained in a matrix format.

1.2 Assumptions

We are making no assumptions.

1.3 Algorithms and Implementation

- Take the matrix as input from user.
- Find minimum element in each row and store these in set s1.
 - Go to each row and take first element as the minimum element.

- Start comparing this *min* element with the other elements in the row.
If the **element** is smaller than *min* then
assign this **element** to *min*
- Find maximum element in each column and store these in set s2.
 - Go to each column and take first element as the maximum element.
 - Start comparing this *max* element with the other elements in the row.
If the **element** is greater than *max* then
assign this **element** to *max*
- Combine sets s1 and s2 to form set s.
- Find prime numbers in set s and display them.
- Create a string from the numbers in set s.
- Ask user for substring to search for.
- Find first index for occurrence of substring using windowing technique.
- If substring does not exist, display not found.
- Find local minima and local maxima in the set s. Note that first and last element do not come under local minima and local maxima.
- Display the local minima and local maxima as output.

1.4 Input and Output format

- **Input format**
Input is given in the form of an NxN matrix with the order of matrix being given as command line argument
- **Output format**
Output is displayed in the Terminal as per the problem statement.

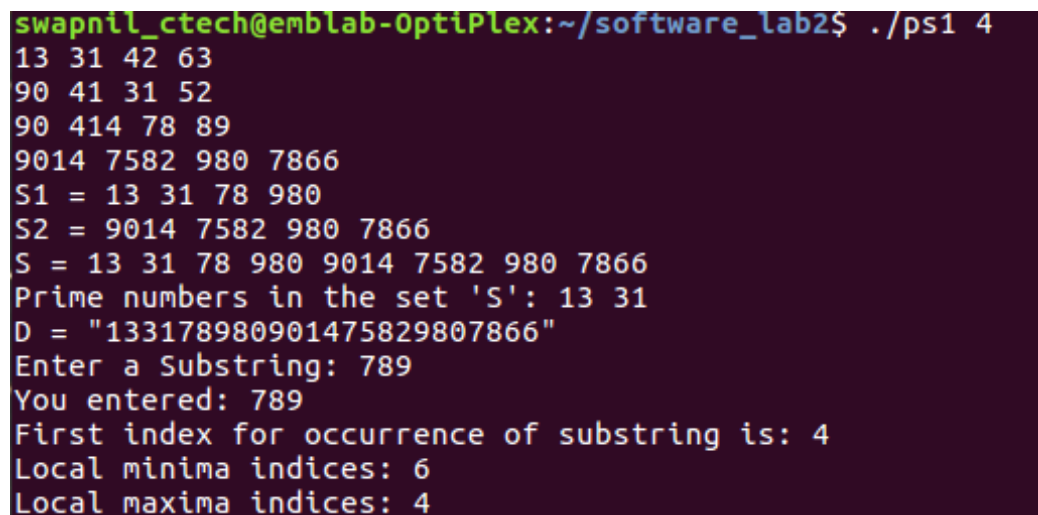
1.5 Test cases

```
./ps1 4
13 31 42 63
90 41 31 52
90 414 78 89
9014 7582 980 7866
```

1.6 Difficulties /Issues faced

- Difficulty faced in finding the presence of substring.

1.7 Screenshots



```
swapnil_ctech@emlab-OptiPlex:~/software_lab2$ ./ps1 4
13 31 42 63
90 41 31 52
90 414 78 89
9014 7582 980 7866
S1 = 13 31 78 980
S2 = 9014 7582 980 7866
S = 13 31 78 980 9014 7582 980 7866
Prime numbers in the set 'S': 13 31
D = "133178980901475829807866"
Enter a Substring: 789
You entered: 789
First index for occurrence of substring is: 4
Local minima indices: 6
Local maxima indices: 4
```

Figure 1: Output of program 1

2 Problem Statement 2

2.1 Problem statement

Split it Wisely! Jeevan, Ramesh, Manisha, Parth and little chhaya went for a trip to Rishikesh. After returning from a trip, they are trying hard to split the expenses equally and settle their debts to each other. Help them splitting the expenses so that everyone gets to know how much money they owe to each other.

We have to write a program for splitting the amount among people.

2.2 Assumptions

- Name of the people is of max. length 100.
- The maximum length of purpose is 100.

2.3 Algorithms and Implementation

- Create a structure to hold the information about members.
- Take the name of members as input.
- Take the expense information from user as input.
- Ask user for choice of spltting.
- Split the amount based on the user's choice.

2.4 Input and Output format

- **Input format**
Input is given in the form the names of members and then the expenses information and then the choice of method to split the money.
- **Output format**
Output is shown in the Terminal as per problem statement.

2.5 Test cases

```
./ps2 4
akash
swap
aman
deepak
akash 2000 hotel
akash 1000 min
deepak 0 pizza
swap 200 cap
done
4
2 1 1 0
```

2.6 Difficulties /Issues faced

- Creating the structure to store data.
- Calculating shares of each member.
- Determining whether float is required for amount.

2.7 Screenshots

```
swapnil_ctech@emblab-OptiPlex:~/software_lab2$ ./ps2 4
Input the names of Members:
akash
swap
aman
deepak
Members added.

Start adding the expenses:
akash 2000 hotel
akash 1000 travel
swap 200 caps
done
All Expenses Added!Choose your option:
1. Split equally
2. Split unequally
3. Split by percentages
4. Split by shares
1

Final Calculations are
akash will get back:    Rs. 2200.00
swap has to pay:       Rs. 600.00
aman has to pay:       Rs. 800.00
deepak has to pay:     Rs. 800.00
```

Figure 2: Output of program 2 with choice 1

```

swapnil_ctech@emlab-OptiPlex:~/software_lab2$ ./ps2 4
Input the names of Members:
akash
swap
aman
deepak
Members added.

Start adding the expenses:
akash 1200 hotel
akash 100 caps
swap 800 travel
deepak 1800 pizza
done
All Expenses Added!Choose your option:
1. Split equally
2. Split unequally
3. Split by percentages
4. Split by shares
2
Enter contributions to total amount in terms of amount that has to be paid by akash swap
aman deepak : 1000 1000 1000 900

Final Calculations are
akash will get back:      Rs. 300.00
swap has to pay:         Rs. 200.00
aman has to pay:         Rs. 1000.00
deepak will get back:    Rs. 900.00

```

Figure 3: Output of program 2 with choice 2

```

swapnil_ctech@emlab-OptiPlex:~/software_lab2$ ./ps2 4
Input the names of Members:
akash
swap
aman
deepak
Members added.

Start adding the expenses:
akash 1000 burger
swap 2500 travel
aman 9000 hotel
done
All Expenses Added!Choose your option:
1. Split equally
2. Split unequally
3. Split by percentages
4. Split by shares
3
Enter contributions to total amount in terms of percentage that has to be paid by akash swap aman deepak : 25 25 40 60
Error!!!
Please enter the share percentages again!!
Enter contributions to total amount in terms of percentage that has to be paid by akash swap aman deepak : 25 25 20 30

Final Calculations are
akash has to pay:      Rs. 2125.00
swap has to pay:      Rs. 625.00
aman will get back:    Rs. 6500.00
deepak has to pay:     Rs. 3750.00

```

Figure 4: Output of program 2 with choice 3


```

swapnil_ctech@emlab-OptiPlex:~/software_lab2$ ./ps2 4
Input the names of Members:
akash
swap
aman
deepak
Members added.

Start adding the expenses:
aman 1000 burger
swao 9000 travel
Invalid member
Enter again:
swap 9000 travel
aman 1200 pizza
done
All Expenses Added!Choose your option:
1. Split equally
2. Split unequally
3. Split by percentages
4. Split by shares
4
Enter contributions to total amount in terms of share for akash swap aman deepak : 2 1 1 0

Final Calculations are
akash has to pay:      Rs. 5600.00
swap will get back:    Rs. 6200.00
aman has to pay:       Rs. 600.00
deepak has to pay:     Rs. 0.00

```

Figure 5: Output of program 2 with choice 4

3 Appendix

3.1 Appendix A: Code for ps1

```
#include<stdio.h>
#include<stdlib.h>// atoi
#include<string.h>//strcat

typedef unsigned int ui;
ui* findMinInRow(ui *arr,int n);
ui* findMaxInCol(ui *arr,int n);
ui* combineS1S2(ui* s1, ui* s2, int n);
ui isPrime(ui num);
int findSubStr(char* input_str, char* str);

int main(int argc, char* argv[])
{

if(argc!=2)
{
printf("Wrong format!!\nPlease enter ./ps1 [size]\n");
return -1;
}

int n = atoi(argv[1]);
ui *A;
char str[1024]={};
char tmp_str[100];
char input_str[1024];
int firstIdx;

A = (ui*)malloc(sizeof(ui)*n*n);

for(int i=0; i<n; ++i)
{
for(int j=0; j<n; ++j)
{
scanf("%u", (A+i*n+j));
}
}
```

```

// find min in each row
ui* s1 = findMinInRow((ui*) A,n);
printf("S1 = ");
for(int i=0; i<n; i++)
{
printf("%u ",s1[i]);
}

// find max in each row
ui* s2 = findMaxInCol((ui*) A,n);

printf("\nS2 = ");
for(int i=0; i<n; i++)
{
printf("%u ",s2[i]);
}

// combine s1 and s2
ui* s = combineS1S2(s1,s2,n);

printf("\nS = ");
for(int i=0; i<2*n; i++)
{
printf("%d ",s[i]);
}

// find prime numbers
printf("\nPrime numbers in the set 'S': ");
for(int i=0; i<2*n; i++)
{
if(isPrime(s[i]) == 1)
printf("%d ",s[i]);
}
printf("\n");

// create string from set S
for(int i=0; i<2*n; i++)
{
sprintf(tmp_str,"%u",s[i]);
strcat(str,tmp_str);
}

```

```

}

printf("D = \\");
for(int i=0; str[i]!='\\0'; i++)
printf("%c",str[i]);

printf("\\\\nEnter a Substring: ");
scanf("%s",input_str);
printf("You entered: %s\\n",input_str);

firstIdx = findSubStr(input_str, str);
if(firstIdx != -1)
{
printf("First index for occurrence of substring is: %d\\n",firstIdx);
}
else
{
printf("No substring matched\\n");
}

// find local minima and maxima
int local_minima_indices[n];
int local_maxima_indices[n];
int lmin_idx=0, lmax_idx=0, lmin_flag=0, lmax_flag=0;
for(int i=1; i<2*n-1; i++)
{
if(s[i]<s[i+1] && s[i]<s[i-1])
{
local_minima_indices[lmin_idx] = i;
lmin_flag = 1;
lmin_idx++;
}
else if(s[i]>s[i-1] && s[i]>s[i+1])
{
local_maxima_indices[lmax_idx] = i;
lmax_flag = 1;
lmax_idx++;
}
}

printf("Local minima indices: ");

```

```

if(lmin_flag == 1)
{
for(int i=0; i<lmin_idx; i++)
{
printf("%d ",local_minima_indices[i]);
}
}
else
{
printf("No local minima found!!\n");
}

if(lmax_flag == 1)
{
printf("\nLocal maxima indices: ");
for(int i=0; i<lmax_idx; i++)
{
printf("%d ",local_maxima_indices[i]);
}
}
else
{
printf("No local maxima found!!\n");
}

printf("\n");
return 0;
}

ui* findMinInRow(ui* A,int n)
{

ui *s1 = (ui*) malloc(n*sizeof(ui));
int min;
for(int i=0; i<n; i++)
{
min = *(A+i*n+0);
for(int j=0; j<n; j++)
{
if(*(A+i*n+j) < min)
min = *(A+i*n+j);
}
}
}

```

```

    }
    s1[i] = min;
}

return s1;
}

ui* findMaxInCol(ui* A,int n)
{

    ui *s2 = (ui*) malloc(n*sizeof(ui));
    int max;
    for(int j=0; j<n; j++)
    {
        max = *(A+0+j);
        for(int i=0; i<n; i++)
        {
            if(*(A+i*n+j) > max)
                max = *(A+i*n+j);
        }
        s2[j] = max;
    }

    return s2;
}

ui* combineS1S2(ui* s1, ui* s2,int n)
{
    ui* s = (ui*)malloc(n*2*sizeof(ui));

    int i;
    for(i=0; i<n; i++)
    {
        s[i] = s1[i];
    }

    int j=0;
    while(i<2*n)
    {
        s[i] = s2[j];
        i++;
    }
}

```

```

j++;
}
return s;
}

```

```

ui isPrime(ui num)
{
    if(num <= 1)
        return 0;
    else if(num == 2 || num == 3)
        return 1;
    else
    {
        for(ui i=2; i<num; ++i)
        {
            if(num % i == 0)
                return 0;
        }
    }
    return 1;
}

```

```

int findSubStr(char* input_str, char* str)
{
    int j = 0;
    int idx = -1;
    for(int i=0; i<= (strlen(str) - strlen(input_str)); i++)
    {
        for(j=0; j<strlen(input_str); j++)
        {
            if(str[i+j] != input_str[j])
            {
                break;
            }
        }
        if(j==strlen(input_str))
            idx = i;
    }

    return idx;
}

```

```
}
```

3.2 Appendix B: Code for ps2

```
#include<stdio.h>
#include<stdlib.h>// atoi
#include<string.h>// strcmp

struct members
{
char name[100];
float amt_paid;
char purpose[100];
float amt_to_pay;
float contribution;
float share;
float multiplier;
};

void split_equally(struct members m[],int n);
void split_unequally(struct members m[],int n);

void split_by_perc(struct members m[],int n);
void split_by_shares(struct members m[],int n);

void init_default(struct members *m);
int find_mem_idx(struct members mem[],int n,char mem_name[]);
void disp_all_mem(struct members mem[],int n);
void final_calculations(struct members mem[],int n);

int main(int argc, char* argv[])
{

if(argc != 2)
{
printf("Wrong format!!\nPlease enter ./ps2 [no. of members involved in trip]\n")
return -1;
}
int n = atoi(argv[1]);
struct members mem[n];
```



```

int choice;
printf("Input the names of Members: \n");

for(int i=0; i<n; i++)
{
scanf("%s",mem[i].name);
init_default(&mem[i]);
}
printf("Members added.\n");

printf("\nStart adding the expenses: \n");

int memIdx;
char mem_name[100];
int continue_flag = 0;
scanf("%s",mem_name);
fflush(stdin);

while(strcmp(mem_name,"done") != 0)
{
float amt_paid;
char purpose[100];

scanf("%f ",&amt_paid);
fflush(stdin);
scanf("%s", purpose);

memIdx = find_mem_idx(mem,n,mem_name);
if(memIdx == -1)
{
printf("Invalid member\nEnter again:\n");
continue_flag = 1;
}
else
continue_flag = 0;

if(continue_flag == 0)
{
mem[memIdx].amt_paid += amt_paid;
strcpy(mem[memIdx].purpose,"");
strcpy(mem[memIdx].purpose,purpose);
}
}

```

```

printf("Record entered successfully!!!\n");
}

scanf("%s",mem_name);
}

printf("All Expenses Added!");

printf("Choose your option:\n");
printf("1. Split equally\n2. Split unequally\n3. Split by percentages\n4. Split ");
scanf("%d",&choice);
switch(choice)
{
case 1:
split_equally(mem,n);
break;
case 2:
split_unequally(mem,n);
break;

case 3:
split_by_perc(mem,n);
break;

case 4:
split_by_shares(mem,n);
break;
default:
printf("Invalid option!!");
}
// disp_all_mem(mem,n);

printf("\nFinal Calculations are\n");
final_calculations(mem,n);
printf("\n");

return 0;
}

void init_default(struct members *m)
{

```

```

m->amt_paid = 0.0;
m->amt_to_pay = 0.0;
strcpy(m->purpose,"");
m->contribution = 0.0;
}

int find_mem_idx(struct members mem[],int n,char mem_name[])
{
for(int i=0; i<n; i++)
{
if(strcmp(mem[i].name,mem_name) == 0)
return i;
}
return -1;
}

void split_equally(struct members m[],int n)
{
float total = 0.0;
for(int i=0; i<n; i++)
{
total += m[i].amt_paid;
}
float split = total/n;
// printf("split: %0.2f\n",split);
for(int i=0; i<n; i++)
{
m[i].amt_to_pay = split - m[i].amt_paid;
// printf("have to pay: %f\n",m[i].amt_to_pay);
}

// disp_all_mem(m,n);

}

void split_unequally(struct members m[],int n)
{
printf("Enter contributions to total amount in terms of amount that has to be pa
for(int i=0; i<n; i++)
{
printf("%s ",m[i].name);
}
printf(": ");

```

```

float contri_sum = 0.0;
for(int i=0; i<n; i++)
{
scanf("%f",&m[i].contribution);
contri_sum += m[i].contribution;
}

float total = 0.0;
for(int i=0; i<n; i++)
{
total += m[i].amt_paid;
}
if(contri_sum != total)
{
printf("Error!!!\nPlease enter the amounts again!!\n");
split_unequally(m,n);
}
else
{
for(int i=0; i<n; i++)
{
m[i].amt_to_pay = m[i].contribution - m[i].amt_paid;
}
}
// disp_all_mem(m,n);

}

void split_by_perc(struct members m[],int n)
{
printf("Enter contributions to total amount in terms of percentage that has to b
for(int i=0; i<n; i++)
{
printf("%s ",m[i].name);
}
printf(": ");

float total_perc = 0.0;
for(int i=0; i<n; i++)
{
scanf("%f",&m[i].share);
total_perc += m[i].share;
}

```

```

}
if(total_perc != 100)
{
printf("Error!!!\nPlease enter the share percentages again!!\n");
split_by_perc(m,n);
}

float total = 0.0;
for(int i=0; i<n; i++)
{
total += m[i].amt_paid;
}
for(int i=0; i<n; i++)
{
m[i].amt_to_pay = m[i].share * total/100 - m[i].amt_paid;
}

}
void split_by_shares(struct members m[],int n)
{
printf("Enter contributions to total amount in terms of share for ");
for(int i=0; i<n; i++)
{
printf("%s ",m[i].name);
}
printf(": ");
for(int i=0; i<n; i++)
{
scanf("%f",&m[i].multiplier);
}
float total = 0.0;
for(int i=0; i<n; i++)
{
total += m[i].amt_paid;
}
float split = total/n;
for(int i=0; i<n; i++)
{
m[i].amt_to_pay = (split*m[i].multiplier - m[i].amt_paid);
printf("have to pay: %f\n",m[i].amt_to_pay);
}
}

```

```
}
```

```
void disp_all_mem(struct members mem[],int n)
```

```
{
```

```
for(int i=0; i<n; i++)
```

```
{
```

```
printf("%s %0.2f %s %0.2f\n",mem[i].name, mem[i].amt_paid, mem[i].purpose,mem[i]
```

```
}
```

```
}
```

```
void final_calculations(struct members mem[],int n)
```

```
{
```

```
for(int i=0; i<n; i++)
```

```
{
```

```
if(mem[i].amt_to_pay < 0)
```

```
{
```

```
printf("%s will get back:\tRs. %0.2f\n",mem[i].name,mem[i].amt_to_pay*-1);
```

```
}
```

```
else
```

```
{
```

```
printf("%s has to pay:\tRs. %0.2f\n",mem[i].name,mem[i].amt_to_pay);
```

```
}
```

```
}
```

```
}
```

References

- [1] “C programming tutorial.” <https://www.tutorialspoint.com/cprogramming/index.htm>. Accessed on 2019-08-06.
- [2] “C programming language - geeksforgeeks.” <https://www.geeksforgeeks.org/c-programming-language/>. Accessed on 2019-08-06.