LAB NO: 1 **Date:**

REVIEW OF C PROGRAMMING CONCEPTS: ARRAYS AND FUNCTIONS

Objectives:

In this lab, student will be able to:

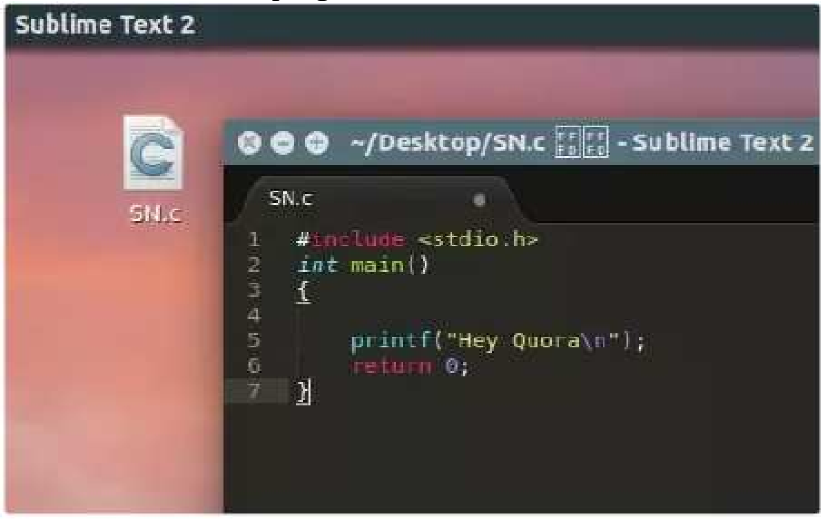
- Review C arrays and functions
- Familiarize with sublime editor for writing C programs
- Compile, execute and debug C program using ddd

I. SUBLIME EDITOR QUICK HELP GUIDE

Creating a source file:

1. Login to student account in Ubuntu
2. Press CNTRL + ALT+ T to open the *terminal*. Alternatively, choose *terminal* from dash home by typing the query 'terminal'

   EX: **user@user:~$** subl
3. Open sublime editor, write a program and save it with .c extension.



```c
#include <stdio.h>
int main()
{

    printf("Hey Quora\n");
    return 0;
}
```

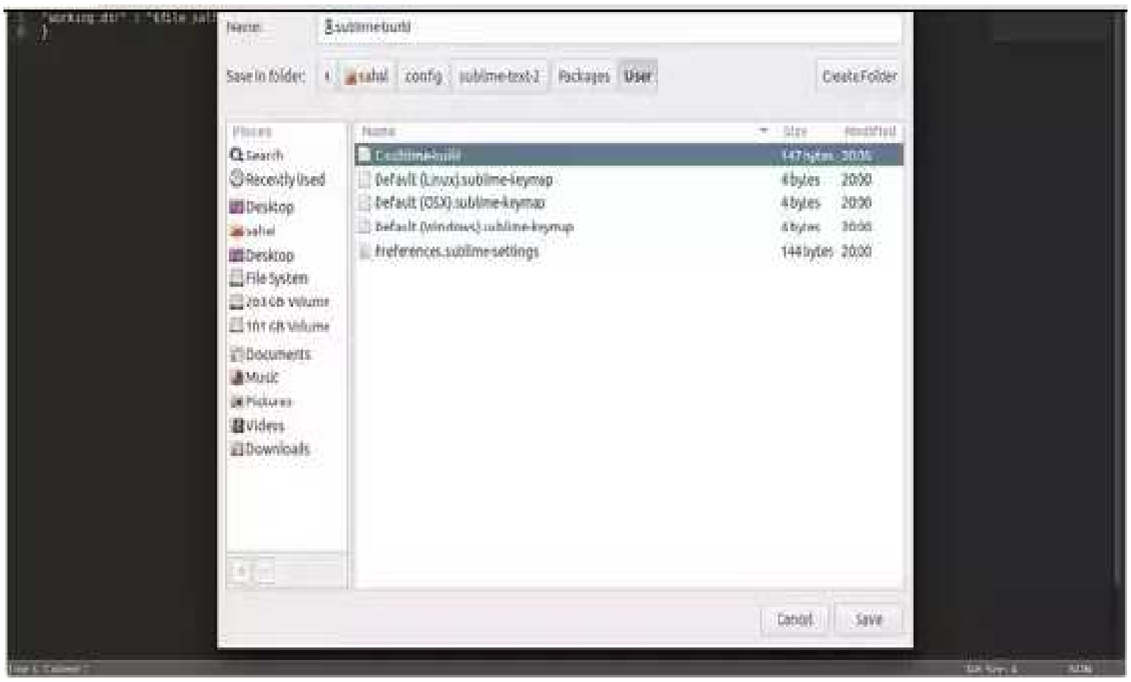
4. Make new c build in sublime editor.
   Tools→build system→newbuild system

5. Paste this code in it:

Open Sublime Editor, write a program and save it with .c extension.

```
{
"cmd" : ["gcc", "$file_name", "-o", "${file_base_name}", "-lm", "-Wall"],
"selector" : "source.c",
"shell":false,
"working_dir" : "$file_path"
}
```

Save it as C.sublime-build.



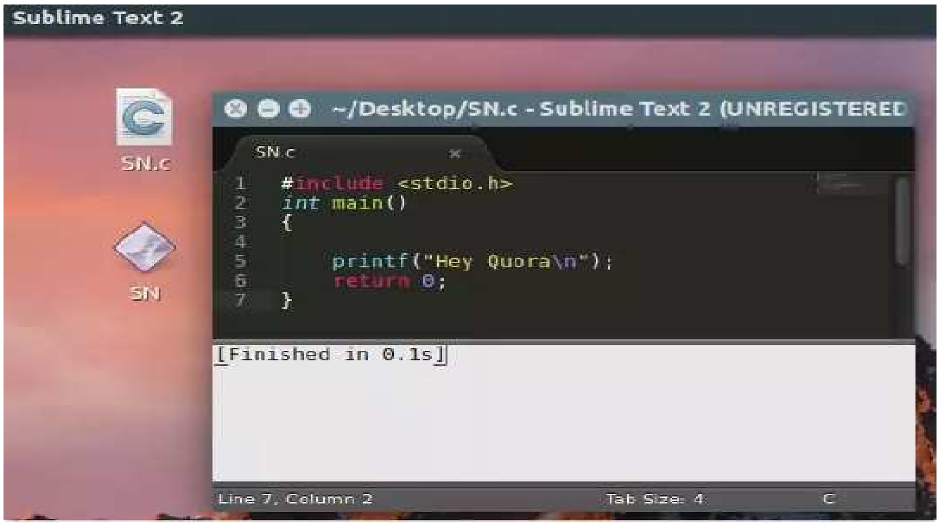
Name: C.sublime-build

Save in folder: sahal / config / sublime-text-2 / Packages / User

```
Name                                     Size       Modified
C.sublime-build                          147 bytes  20:06
Default (Linux).sublime-keymap           4 bytes    20:00
Default (OSX).sublime-keymap             4 bytes    20:00
Default (Windows).sublime-keymap         4 bytes    20:00
Preferences.sublime-settings             144 bytes  20:00
```

6. Select C build system.
   Tools→Build system→C

7. Build your C program.
   Tools→Build (Ctrl + B)

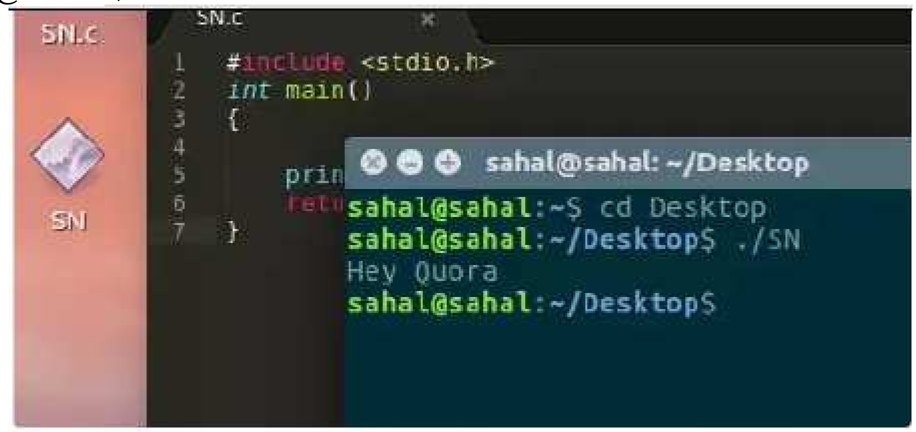


```c
#include <stdio.h>
int main()
{

    printf("Hey Quora\n");
    return 0;
}
```

```
[Finished in 0.1s]
```

8. Open terminal and locate your C program.
   Ex: **user@user:~$** cd Desktop
9. Run build file by typing:
   user@user:~$ ./SN



```
sahal@sahal:~$ cd Desktop
sahal@sahal:~/Desktop$ ./SN
Hey Quora
sahal@sahal:~/Desktop$
```

OR

Compiling and Executing through terminal:

- Type the Compile command as: cc -g –o *outputfile inputcfile*
  - Ex:
    - cc –o factorial factorial.c

      In this case the executable file is created as *factorial*. To run the executable go to terminal and type ./factorial. (Here ./

refers to the current directory, alternatively we can specify the absolute or relative path of the executable file).

- cc factorial.c
  In this case the executable file is created as *a.out* by default. Remember this executable file will be over written when you run the same command again.
  To run the executable go to terminal and type ./a.out

Debugging through data display debugger:

- Open the terminal and run the ddd commands (refer ddd video)

II. SOLVED EXERCISE:

1) Write a C program to sort given list of n integers into ascending order using selection sort. Use function to sort.

Description: Assume we have an array ‘A’ with ‘N’ number of elements. This algorithm arranges elements in ascending order. ‘Pass’ is an index variable, which indicates the number of passes. The variable ‘min_index’ denotes the position of the smallest element encountered in that pass. ‘I’ is another index variable. The array and the size are passed to the function.

Algorithm: Selection Sort

Step 1: Using Pass index variable repeat steps from first record to last – 1 records and perform all the steps up to 4.

Step 2: Initialize the minimum index as

```
        min_index = pass.
```

Step3: Obtain the element with the smallest value.

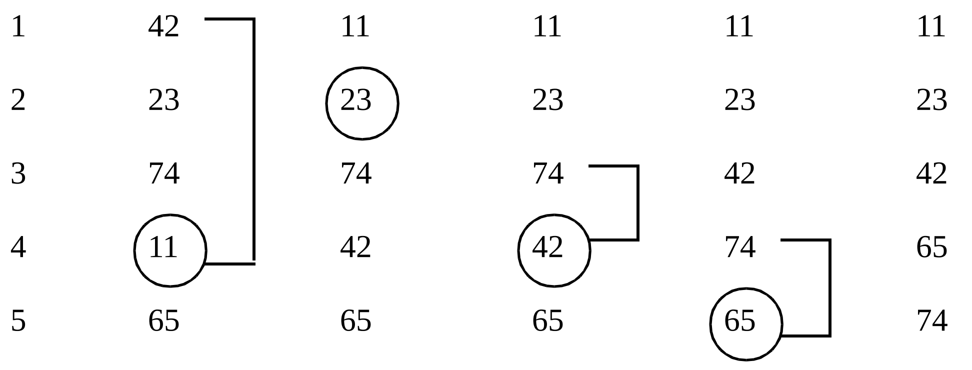
```
      for(i= pass+1; i < N; i++)
      {
        if( A[i] < A [min_index])
                min_index = i;
      }
```

Step4: Exchange the elements

```
      if(min_index != pass)
      {
          temp =  A[pass];
            A[pass] = A[min_index];
          A[min_index] = temp;
      }
```

Step5: Stop

Trace of Selection Sort:



1	42	11	11	11	11
2	23	23	23	23	23
3	74	74	74	42	42
4	11	42	42	74	65
5	65	65	65	65	74

File Name: selection_sort_fun.h

```
/*  function to find the minimum value */
int findmin(int b[10], int k)
{
   int min = 0, j;
   for (j = 1; j <= k; j++)
   {
      if (b[min] < b[j])
         min = j;
    }
   return(min);
}
/*  function to exchange the  value */
void exchange(int b[10], int k)
{
   int  temp, small, j;
   for (j = k - 1; j >= 1; j--)
   {
      small = findmin(b, j);
      temp = b[small];
      b[small] = b[j];
      b[j] = temp;
   }
   return;
}
```

File Name: selection_sort.c

```
#include <stdio.h>
#include "selection_sort_fun.h"
void main()
{
   int array[10];
   int i, j, n, temp;
   printf("Enter the value of n \n");
   scanf("%d", &n);
   printf("Enter the elements  \n");
   for (i = 0; i < n; i++)
      scanf("%d", &array[i]);
   /*  Selection sorting begins */
```

```
        exchange(array, n);
        printf("The sorted list is (using selection sort): \n");
        for (i = 0; i < n; i++)
            printf("%d\t", array[i]);
        }
```

Sample Input and output:

```
Enter the total no of elements: 5
Enter the elements: 99 20 -12 43 34
The sorted list is (using selection sort):  -12  20  34  43  99
```

III. LAB EXERCISE:

Write a menu-driven program in C to:

1) Find sum of given n numbers using a 1D array using a function.
2) Implement Linear Search on list of integers.
3) Find sum of 2 matrices using a function named add with suitable parameters.
4) Find biggest in a list of numbers using a function.

IV. ADDITIONAL EXERCISES:

1) **Random number generation and finding the frequency of occurrence**: Generate a large number of random numbers (say around 10K Samples). Each sample value should be between -100 to 100 (integers only). After generating the samples find the frequency of each distinct sample. Repeat the above steps with unknown size (hint: user will decide at run time). The purpose is to realize the advantages and disadvantages of using array.
2) **Addition of polynomials with two terms:** To perform different operations on polynomial with two terms x, y using 2-D array representations. Operations like addition and multiplication have to be implemented [ref: J.P Trembly]. If the 2D array representation is sparse then optimize the memory usage by using suitable alternative representation.

[OBSERVATION SPACE – LAB 1]

[OBSERVATION SPACE – LAB 1]

[OBSERVATION SPACE – LAB 1]

[OBSERVATION SPACE – LAB 1]

[OBSERVATION SPACE – LAB 1]

[OBSERVATION SPACE – LAB 1]

LAB NO: 2 **Date:**

REVIEW OF C PROGRAMMING CONCEPTS: STRUCTURES AND POINTERS

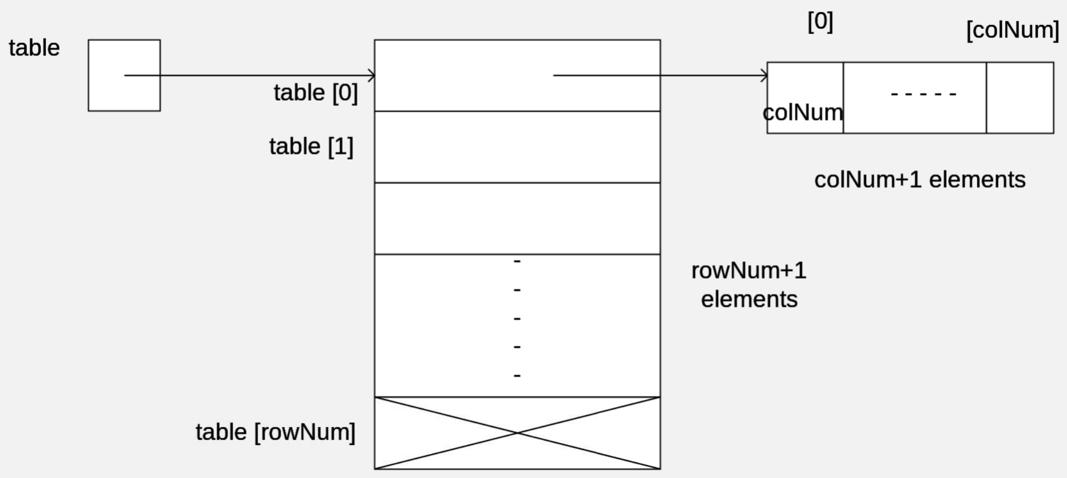
Objectives:

In this lab, student will be able to:
i) Familiarize with syntax and usage of structures and/or pointers
ii) Write C programs making use of structures and pointer concepts

I. SOLVED EXERCISE:

1) Write a C program to implement a ragged array dynamically.

Description: In a ragged array the *table* pointer points to the first pointer in an array of pointers. Each array pointer points to a second array of integers, the first element of which is the number of elements in the list. A sample ragged array structure is shown below.



Algorithm: Construct a ragged array

```
Step 1: Declare a ragged array as a variable table.
Step 2: Ask the user for row size and set a variable – rowNum
Step 3: Allocate space for (rowNum+1) pointers as row pointers. The last row pointer will hold NULL
Step 4: Ask the user for column size and set a variable – colNum
Step 5: Allocate space for (colNum+1) data elements. The first element will hold value contained in colNum itself.
Step 6: Repeat step 3 for all rows
Step 7 : Display ragged array contents.
Step 8: Stop
```

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int rowNum, colNum, i, j;
int **table;
printf("\n enter the number of rows \n");
scanf("%d", &rowNum);
table = (int **) calloc(rowNum+1, sizeof(int *));
for (i = 0; i < rowNum; i++)  /* this will tell which row we are in */
{
        printf("enter size of %d row", i+1);
        scanf("%d", &colNum);
```

```
          table[i] = (int *) calloc(colNum+1, sizeof(int));
              printf("\n enter %d row elements ", i+1);
                         for (j = 1; j <= colNum;  j++)
                           {
                             scanf("%d", &table[i][j]);
                            }
          table[i][0] = colNum;
          printf("size of row number [%d] = %d", i+1, table[i][0]);


}
table[i] = NULL;
for (i = 0; i < rowNum; i++) /* this will tell which row we are in */
{
          printf("displaying %d row elements\n", i+1);
                  for (j = 0; j <= *table[i];  j++)
                                   {
                              printf("%5d", table[i][j]);
                                  }
  printf("\n");
  }
  return 0;
}
```

Sample input and output:

```
enter the number of rows: 3

enter size of row 1: 4
enter row 1 elements: 10 11 12 13

enter size of row 2: 5

enter row 2 elements: 20 21 22 23 24

enter size of row 3

enter row 3 elements: 30 31 32

displaying

10 11 12 13

20 21 22 23 24

30 31 32
```

II. LAB EXERCISES :

Note: Pass parameters using pointer to all the following functions and implement menu-driven program.

1) Write a program in C to add two numbers using pointers.
2) Write a program in C to
   a) Demonstrate passing pointers to a function.
   b) Demonstrate Returning pointer from a function.
   c) Using pointer to pointer.
3) Write a program in C to find roots of quadratic equation as follows.
   a) Write a function called getdata to read coefficients.
   b) Write a function called quadratic to find roots.
   c) Write a function print_roots to print the roots.
   d) Write a main function to invoke all this.
4) Write a C program to print an array in forward direction by adding 1 to pointer and in backward direction by subtracting one from pointer.

III. ADDITIONAL EXERCISES:

1) Write a program to sort an array of student structures according to the roll number.
2) Implement Complex numbers using structures. Write functions to add, multiply, subtract two complex numbers.
3) Implement problem 1 by using structure pointers.
3) Write a C program to demonstrate

a. Unions b. Unions in structures. c. Nested structures.

4) Processing of a BMP file: A BMP file is made out of a header part and the pixels. The header contains basically 2 records with info about width, height etc. These are the header record layouts:

```
typedef struct {
  unsigned short type;
  unsigned long size;
  unsigned short reserved1;
  unsigned short reserved2;
  unsigned long offsetbits;
} BITMAPFILEHEADER;

typedef struct {
  unsigned long size;
  unsigned long width;
  unsigned long height;
  unsigned short planes;
  unsigned short bitcount;
  unsigned long compression;
  unsigned long sizeimage;
  long xpelspermeter;
  long ypelspermeter;
  unsigned long colorsused;
  unsigned long colorsimportant;
} BITMAPINFOHEADER;
```

After these headers the pixels are written:

```
typedef struct {
  unsigned char blue;
  unsigned char green;
```

```
        unsigned char red;
    } SINGLE_PIXEL;
```

Write a program to read a BMP file and invert each RGB pixel by subtracting the read value from 255. Write back the result onto a file with the same header and data.

[Hint: make use of fread() and fwrite() functions by referring to Appendix section]

[OBSERVATION SPACE – LAB 2]

[OBSERVATION SPACE – LAB 2]