Experiment-1

AIM: Write a program to write to a text file and then read it character wise while counting the number of alphabets, numeric characters, special characters, words, spaces and lines.

THEORY:

An fstream object is created and is attached to the file in out mode i.e. the write mode. The string to be written is taken as input from the user. This string is then written to the file. Then the file is opened in the in mode i.e. read mode and is read character by character till end of the file is detected. And by looping the number of alphabets, numeric characters, special symbols, spaces, words are counted. The output is then displayed.

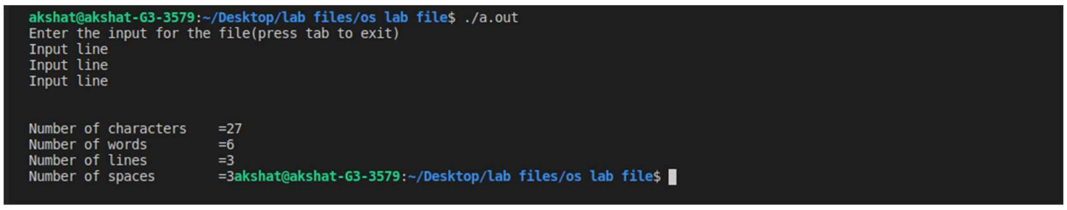
ALGORITHM:

```
Step 1: START
Step 2: fstream object is created and is attached to the file in out mode.
Step 3: Input string is taken from the user.
Step 4: String is written to the file.
Step 5: File is closed.
Step 6: File is opened in the 'in' mode and is read character wise iteratively till eof is detected.
Step 7: Each character is tested for its ASCII value and then appropriate counters are incremented.
Step 8: Final value of all the counters is displayed.
Step 9: END
```

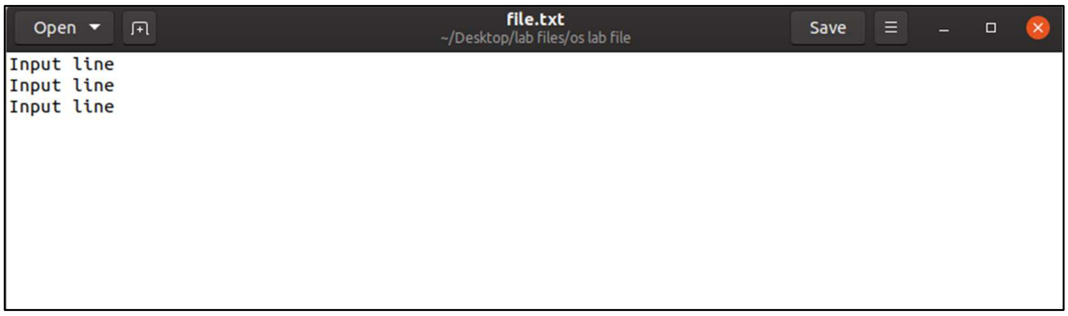
CODE:

```
#include<iostream>
#include<fstream>

using namespace std;

int main(){
    char input[1000];
    ofstream fout("file.txt");
    cout<<"Enter the input for the file(press tab to exit)\n";
    cin.getline(input, 1000, '\t');
    fout<<input;
    fout.close();
    ifstream fin("file.txt");
    int characters, words, lines, spaces;
    characters = words = lines = spaces = 0;
    while(!fin.eof()){
        fin.getline(input, 1000, '\n');
        lines++;
        int i = 0;
        while(input[i] != '\0'){
            if(input[i] != ' ')
                characters++;
            else
                spaces++;
            i++;
        }
    }
    words = spaces + lines;
    fin.close();
    cout<<"\n\nNumber of characters\t="<<characters;
    cout<<"\nNumber of words\t\t="<<words;
    cout<<"\nNumber of lines\t\t="<<lines;
    cout<<"\nNumber of spaces\t="<<spaces;
    return 0;
}
```

OUTPUT:



```
akshat@akshat-G3-3579:~/Desktop/lab files/os lab file$ ./a.out
Enter the input for the file(press tab to exit)
Input line
Input line
Input line


Number of characters    =27
Number of words         =6
Number of lines         =3
Number of spaces        =3akshat@akshat-G3-3579:~/Desktop/lab files/os lab file$
```



```
Input line
Input line
Input line
```

RESULT:

We have successfully counted the number of alphabets, numbers, special characters, spaces in the file as input by the user using file manipulation functions included in the header file fstream.

CONCLUSION:

In this experiment we utilized various inbuilt functions and objects of the fstream header file in order to write some data into a file. After that we further read the data back from the file and then processed the data to determine the count of words, lines, spaces and characters in the file. Through this exercise we learned about various file handling operations in C++.

Experiment-2

AIM: Write a program to implement First Come First Serve scheduling algorithm and calculate the values of waiting time, completion time and turnaround time for the respective arrival times and burst times.

THEORY:

First Come First Serve or FCFS is an example of a scheduling algorithm used by the operating system in order to determine the order of execution of the different processes. In FCFS algorithm the operating system determines the order of processing of a task on the basis of the time of arrival of the various processes. The process with the least or the most recent time of arrival is given precedence over the other processes and is processed first.

With this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the rail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is the removed form the queue. One negative part about utilizing this algorithm is that the average waiting time under this algorithm is often quite long.

ALGORITHM:

Step 1: START

Step 2: Take the number of processes, arrival times and the burst time for the various processes as input.

Step 3: Repeat steps 4, 5 and 6 while value of j is between 0 and number of processes to be processed.

Step 4: The processes with the minimum arrival time which hasn’t already been processed is determined form the input arrival times.

Step 5: The selected arrival time along with its corresponding burst time is used to determine the values for the waiting time, completion time and the turnaround time using appropriate formulae.

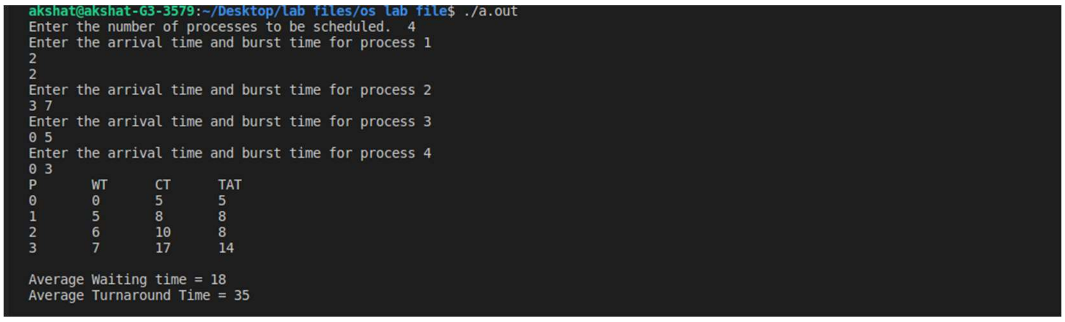
Step 6: The process number, waiting time, completion time and the turnaround time is displayed.

Step 7: END

CODE:

```
#include<iostream>
#include<bits/stdc++.h>

using namespace std;

void calculate(int n, int *AT, int *BT){
    int pos = 0;
    vector<int>visited;
    int WT = 0, CT = 0, TAT = 0, avg_wt = 0, avg_tat = 0;
    int start = 0;
    cout<<"P\tWT\tCT\tTAT\n";
    for(int j = 0; j < n; j++){
        int min = INT16_MAX;
        for(int i = 0; i < n; i++){
            if(AT[i] < min && find(visited.begin(), visited.end(), i) == visited.end()){
                min = AT[i];
                pos = i;
            }
        }
        visited.push_back(pos);
        start = CT;
        WT = (start > min) ? (start - min) : 0;
        CT = start + BT[pos];
        TAT = CT - min;
        avg_wt += WT;
        avg_tat += TAT;
        cout<<j<<"\t"<<WT<<"\t"<<CT<<"\t"<<TAT<<endl;
    }
    cout<<endl<<"Average Waiting time = "<<avg_wt;
    cout<<endl<<"Average Turnaround Time = "<<avg_tat<<endl<<endl;
}

int main(){
    int n;
    cout<<"Enter the number of processes to be scheduled.\t";
    cin>>n;
    int AT[(const int)n], BT[(const int)n];
    for(int i = 0; i < n; i++){
        cout<<"Enter the arrival time and burst time for process "<<i + 1<<endl;
        cin>>AT[i]>>BT[i];
    }
    calculate(n, AT, BT);
    return 0;
}
```

OUTPUT:



```
akshat@akshat-G3-3579:~/Desktop/lab files/os lab file$ ./a.out
Enter the number of processes to be scheduled.  4
Enter the arrival time and burst time for process 1
2
2
Enter the arrival time and burst time for process 2
3 7
Enter the arrival time and burst time for process 3
0 5
Enter the arrival time and burst time for process 4
0 3
P       WT      CT      TAT
0       0       5       5
1       5       8       8
2       6       10      8
3       7       17      14

Average Waiting time = 18
Average Turnaround Time = 35
```

RESULT:

We have successfully determined the values of waiting time, completion time and the turnaround time for the various processes and also the average waiting time and the average turnaround time.

CONCLUSION:

In this experiment we successfully implemented a First Come First Serve scheduling algorithm with the help of arrays that are used to hold the values of the arrival and burst times of the different processes, and finally display the required values for the waiting time, completion time, turnaround time, average waiting time and average turnaround time.