PROBLEM SOLVING

(Solving Various Problems using C Language)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Tummala Sai Padmasree

221710307059

https://github.com/PADMASREE1999/Final-Project

Under the Guidance of

Assistant Professor

Department of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled **“Solving Various Problems using C Language**” to GITAM (Deemed to Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science and Engineering**”. I declare that it was carried out independently by me under the guidance of Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD Tummala Sai Padmasree
Date: 221710307059

GITAM (DEEMED TO BE UNIVERSITY)
Hyderabad-502329, India
Dated:

<u>CERTIFICATE</u>

This is to certify that the Industrial Training Report entitled **“Solving Various Problems Using C Language”** is being submitted by TUMMALA SAI PADMASREE (221710307059) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering** at GITAM (Deemed to Be University), Hyderabad during the academic year 2019-2020.

It is faithful record work carried out by her at the **Computer Science and Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Assistant Professor
Department of CSE

Mr. S. Phani Kumar
Professor and HOD
Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful completion of this internship.

I would like to thank respected **Dr. N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Mr. N. Seetharamaiah,** Principal, GITAM Hyderabad.

I would like to thank respected **Mr. S. Phani Kumar,** Head of the Department of Computer Science and Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Tummala Sai Padmasree
221710307059

TABLE OF CONTENTS

1. Introduction

Problem Solving is the Process of Designing and carrying out certain steps to reach a Solution. Six problems which are listed below are of different complexity and require different approaches and logics in order to achieve desired Output/ Solution.

1. **Zombie World:** In this problem we calculate the remaining energy of Bob using the given formula to check whether he can defeat all the Zombies or not.
2. **Alpha Numeric Sorting:** In this problem we sort the given string and integer arrays and print them in the order of string followed by an integer separated by space.
3. **Prime Fibonacci:** In this problem we need to form unique numbers using all the prime numbers in the given range and taking the count of the prime numbers in the list of unique numbers, maximum and minimum prime number as the first two elements of the Fibonacci series, find the last number of the Fibonacci series.
4. **Super ASCII String Checker:** In this problem we give a new ASCII value to every alphabet like a=1, b=2, …, z=26 and check whether the count of each alphabet in the given string is equal to its new value.
5. **Consecutive Prime Sum:** In this problem we need to find the count of sum of consecutive prime numbers such that the sum is also prime.
6. **Houses Problem**: In this problem we need to find the maximum value stolen by the thief.

I have executed projects in C language. I have used CodeBlocks and GDB Online Debugger to execute the codes.

2. Problem 1
Zombie World

In this problem we calculate the remaining energy of Bob using the given formula to check whether he can defeat all the Zombies or not.

2.1. Problem statement:

One day Bob is playing a Zombie World video game. In the Zombie World game each round will contain N zombie's and each zombie's energy is Zi (where 1<=i<=N). Bob will start the round with B energy. In order to move to the next level. Bob needs to kill all the N zombie's but Bob can select any one among N Zombie's. If the energy of Bob (B) is less than Zombie energy (Zi) then Bob will die and lose the round else Bob will win, during the fighting with the zombie, Bob will lose his energy by (Zi%2) + (Zi/2). At any point of game Bob will play optimally. Now your task is to find out whether Bob can reach the next level or not.

Input Format:

First line contains B and N, separated by space, where B is the energy of Bob and N is the number of Zombies. Next line will contain N spaced integers each will represent the energy of the Zombie.

Constraints:

- 1<=N<=10^4 1<=B<=10^9 1<=Zi<=10^5
- Note: for this problem all the divisions are integer divisions. Output Format: Print "YES" or "NO" depending upon whether Bob can reach the next level or not.
- For Valid Input, print YES or NO
- For Invalid Input, print Invalid Input

Sample Input and Output:

SNo.	Input	Output
1	35 3 5 9 6	YES

2	456 68 a	Invalid Input
3	4 4 1 2 3 4	NO

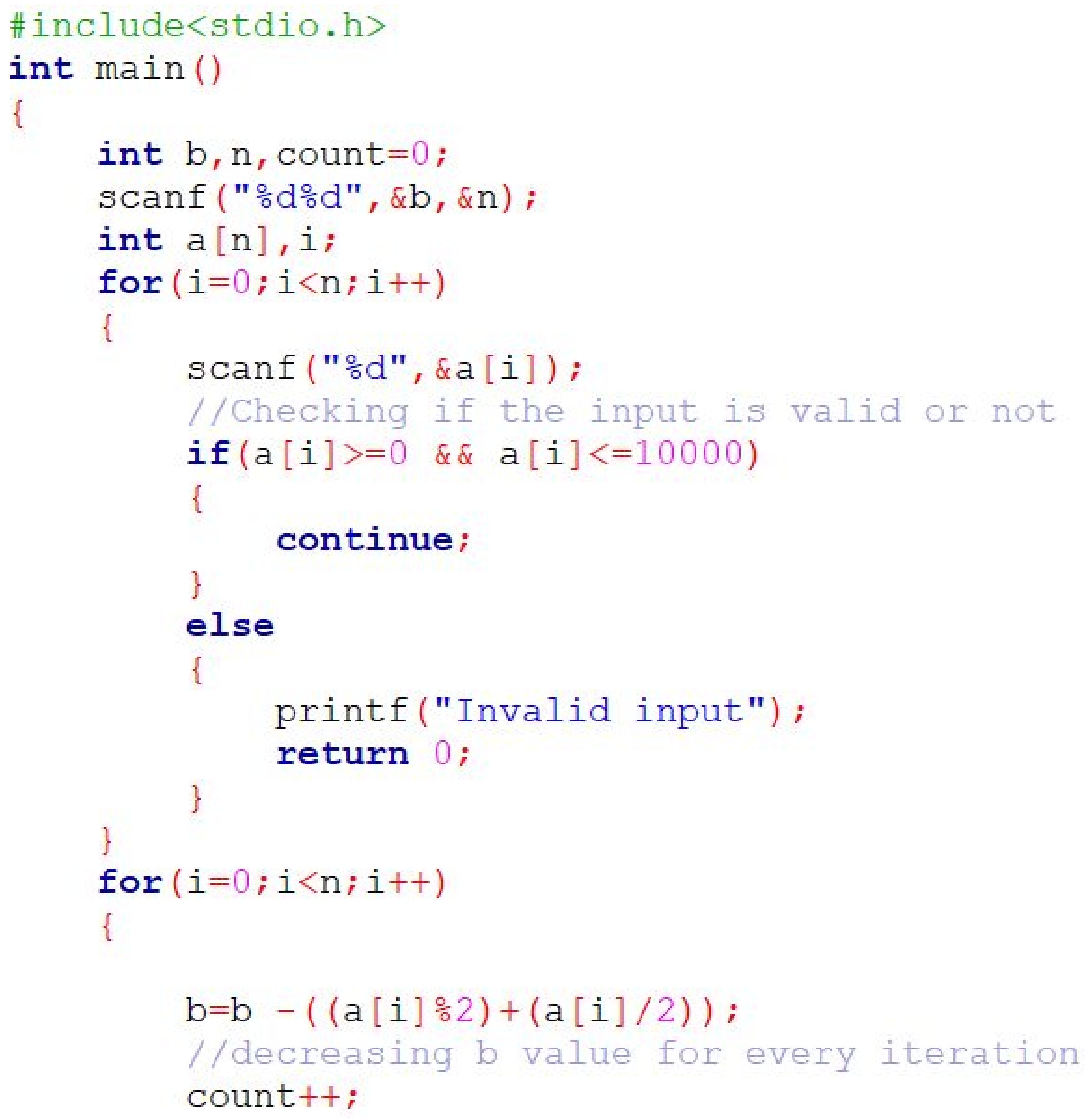
Concepts Used to Solve:

- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
  Syntax:
  The syntax of a for loop in C programming language is −
  ```
  for ( init; condition; increment)
  {
     statement(s);
  }
  ```

- **If-else statement:** If the Boolean expression evaluates to true, then the if block will be executed, otherwise, the else block will be executed. C programming language assumes any non-zero and non-null values as true and if it is either zero or null, then it is assumed as false value.
  ```
  if(boolean_expression)
     {   /* statement(s) will execute if the boolean expression
     is true */}
     else
      {   /*statement(s) will execute if the boolean expression
     is false */}
  ```

- **Arrays:** An array is a collection of data items, all of the same type, accessed using a common name. A one-dimensional array is like a list; A two dimensional array is like a table; The C language places no limits on the number of dimensions in an array, though specific implementations may.

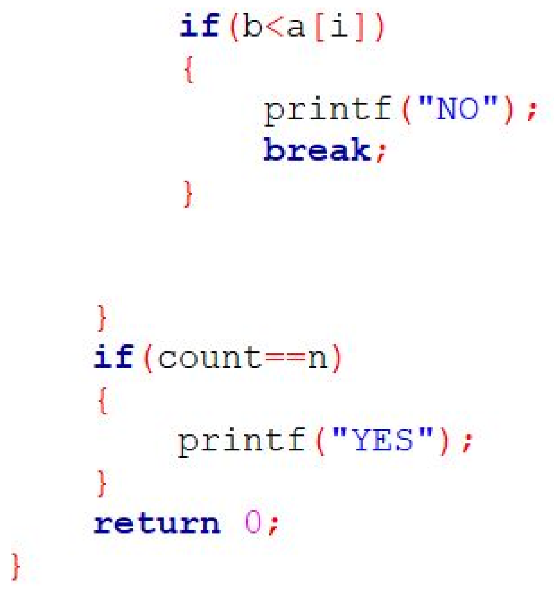
2.2. Coding:



```c
#include<stdio.h>
int main()
{
    int b,n,count=0;
    scanf("%d%d",&b,&n);
    int a[n],i;
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
        //Checking if the input is valid or not
        if(a[i]>=0 && a[i]<=10000)
        {
            continue;
        }
        else
        {
            printf("Invalid input");
            return 0;
        }
    }
    for(i=0;i<n;i++)
    {

        b=b -((a[i]%2)+(a[i]/2));
        //decreasing b value for every iteration
        count++;
```

Fig 2.2.1

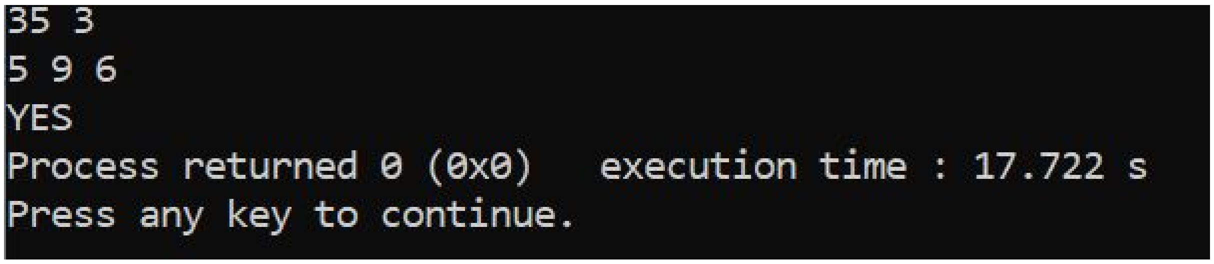


```
        if(b<a[i])
        {
            printf("NO");
            break;
        }


    }
    if(count==n)
    {
        printf("YES");
    }
    return 0;
}
```

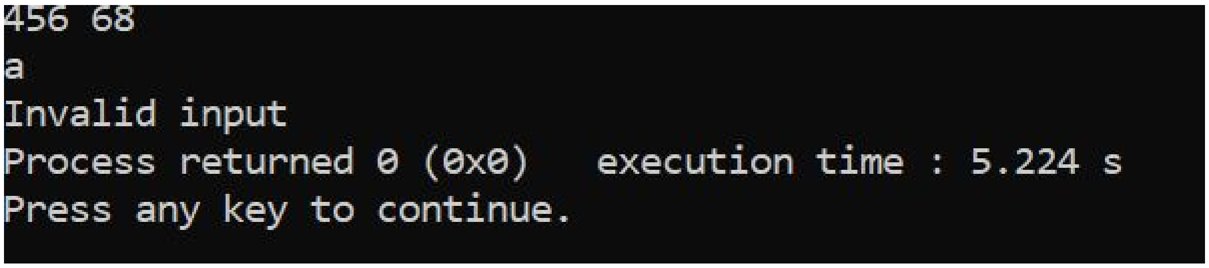
Fig 2.2.2

2.3. Output:



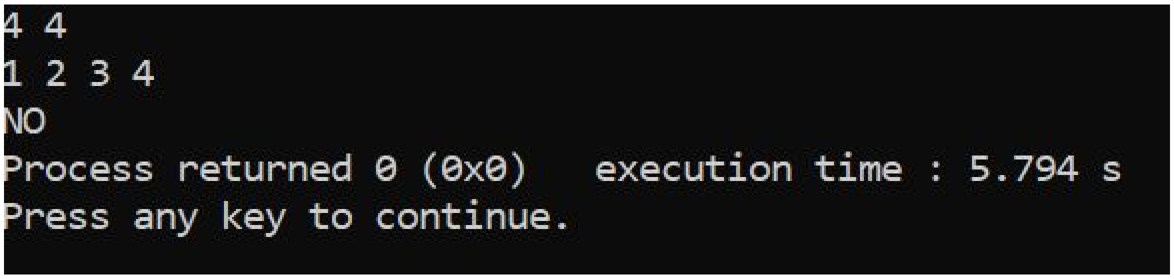
```
35 3
5 9 6
YES
Process returned 0 (0x0)   execution time : 17.722 s
Press any key to continue.
```

Fig 2.3.1



```
456 68
a
Invalid input
Process returned 0 (0x0)   execution time : 5.224 s
Press any key to continue.
```

Fig 2.3.2



```
4 4
1 2 3 4
NO
Process returned 0 (0x0)   execution time : 5.794 s
Press any key to continue.
```

Fig 2.3.3

3. Problem 2
Alpha Numeric Sorting

In this problem we sort the given string and integer arrays and print them in the order of string followed by an integer separated by space.

3.1. Problem Statement:

Given text consisting of words and numbers, sort them both in ascending order and print them in a manner that a word is followed by a number. Words can be in upper or lower case. You have to convert them into lowercase and then sort and print them.

Input Format:

First line contains total number of test cases, denoted by N
Next N lines, each contains a text in which words are at odd position and numbers are at even position and are delimited by space

Output Format:

Words and numbers sorted in ascending order and printed in a manner that a word is followed by a number.

Constraints:

- Text starts with a word
- Count of words and numbers are the same.
- Duplicate elements are not allowed
- Words should be printed in lower case.
- No special characters allowed in text.

Sample Input and Output:-

SNo.	Input	Output
1	2 Sagar 35 sanjay 12 ganesH 53 ramesh 19 Ganesh 59 suresh 19 rakesh 26 laliT 96	 ganesh 12 ramesh 19 sagar 35 sanjay 53 ganesh 19 lalit 26 rakesh 59 suresh 96

Concepts Used to Solve:

- **Function:** A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.
  Defining a function:
  ```
  return_type function_name( parameter list ) {    body of
  the function}
  ```

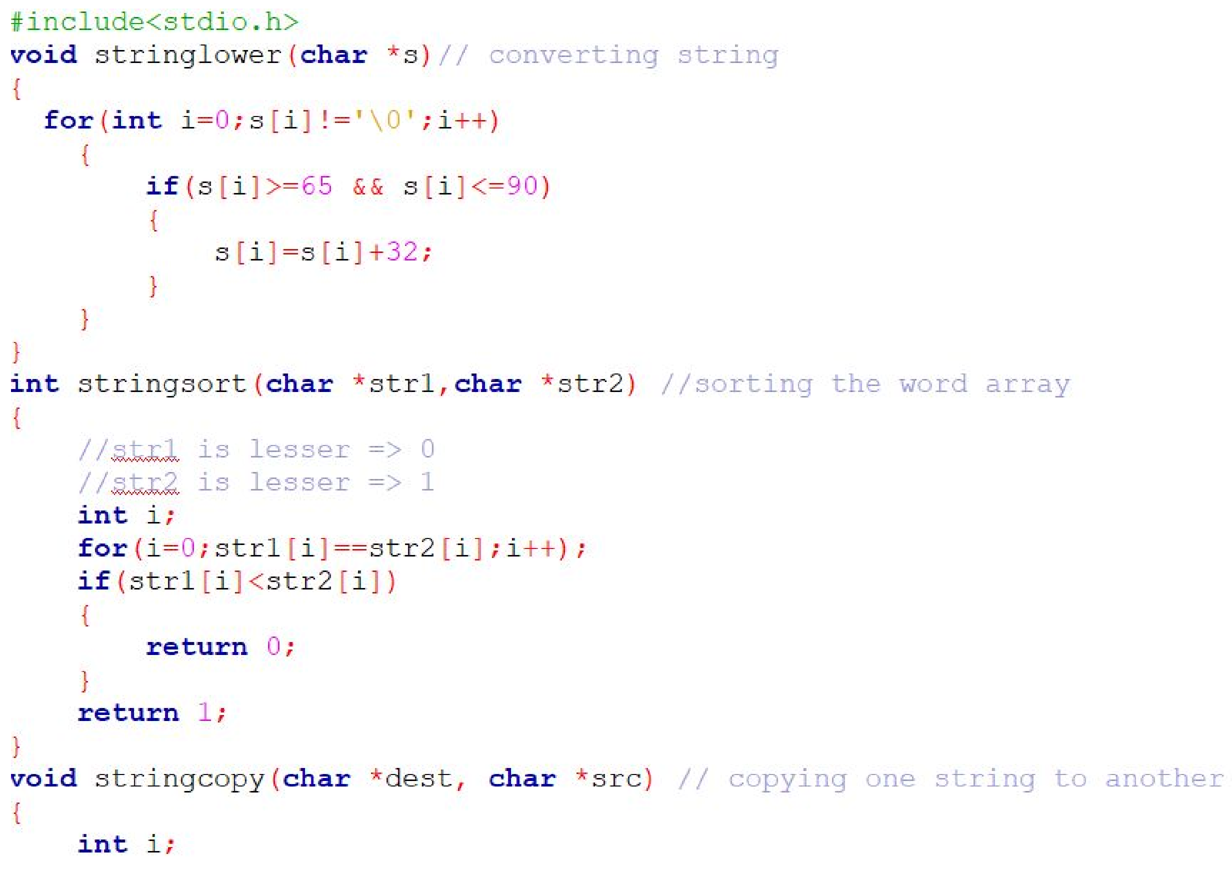
- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
  Syntax:
  The syntax of a for loop in C programming language is −
  ```
  for ( init; condition; increment)
  {
          statement(s);
  }
  ```

- **While loop:** A while loop in C programming repeatedly executes a target statement as long as a given condition is true.
  Syntax
  The syntax of a while loop in C programming language is −
  ```
  while(condition) {   statement(s);}
  ```

- **ASCII:** ASCII is the acronym for the American Standard Code for Information Interchange. It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127.

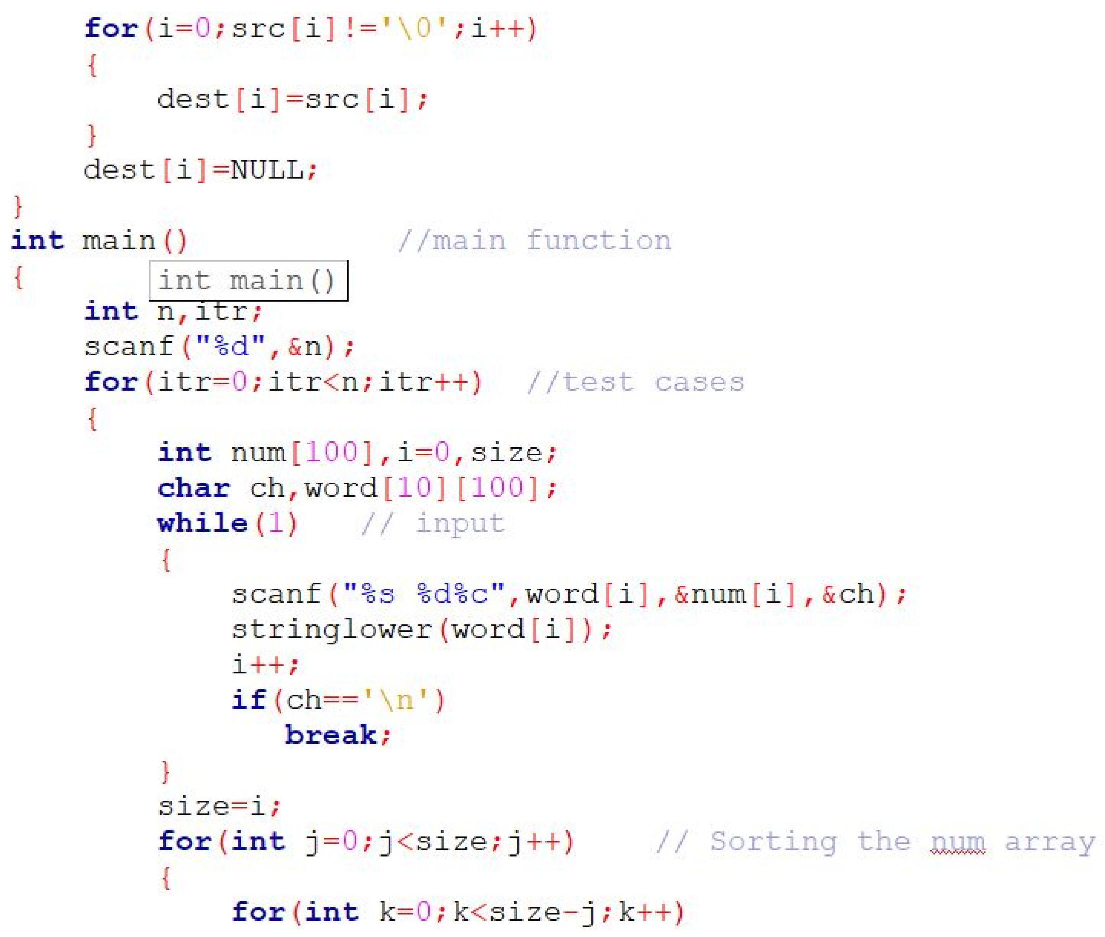
- **Arrays:** An array is a collection of data items, all of the same type, accessed using a common name. A one-dimensional array is like a list; A two dimensional array is like a table; The C language places no limits on the number of dimensions in an array, though specific implementations may.

3.2. Coding:



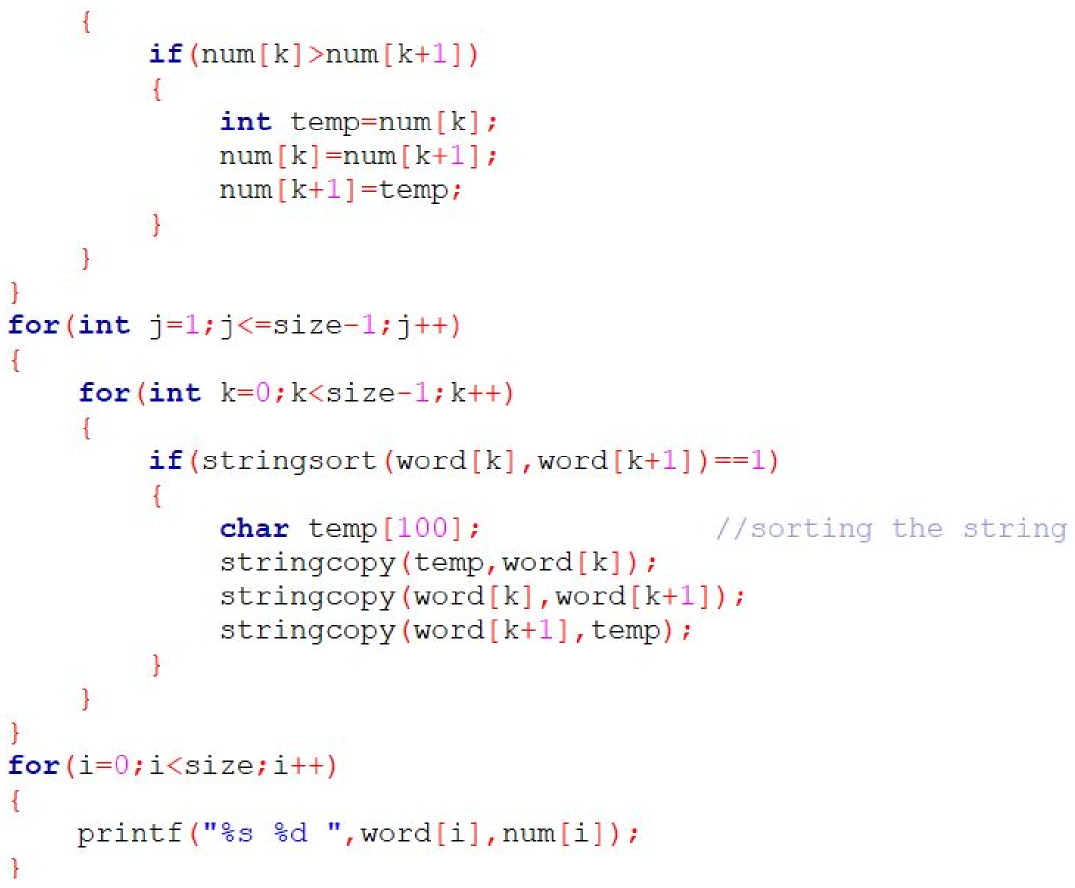
```c
#include<stdio.h>
void stringlower(char *s)// converting string
{
  for(int i=0;s[i]!='\0';i++)
    {
        if(s[i]>=65 && s[i]<=90)
        {
            s[i]=s[i]+32;
        }
    }
}
int stringsort(char *str1,char *str2) //sorting the word array
{
    //str1 is lesser => 0
    //str2 is lesser => 1
    int i;
    for(i=0;str1[i]==str2[i];i++);
    if(str1[i]<str2[i])
    {
        return 0;
    }
    return 1;
}
void stringcopy(char *dest, char *src) // copying one string to another
{
    int i;
```

Fig 3.2.1



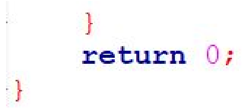
```
    for(i=0;src[i]!='\0';i++)
    {
        dest[i]=src[i];
    }
    dest[i]=NULL;
}
int main()              //main function
{
    int n,itr;
    scanf("%d",&n);
    for(itr=0;itr<n;itr++)  //test cases
    {
        int num[100],i=0,size;
        char ch,word[10][100];
        while(1)    // input
        {
            scanf("%s %d%c",word[i],&num[i],&ch);
            stringlower(word[i]);
            i++;
            if(ch=='\n')
               break;
        }
        size=i;
        for(int j=0;j<size;j++)     // Sorting the num array
        {
            for(int k=0;k<size-j;k++)
```

Fig 3.2.2



```
        {
            if(num[k]>num[k+1])
            {
                int temp=num[k];
                num[k]=num[k+1];
                num[k+1]=temp;
            }
        }
    }
    for(int j=1;j<=size-1;j++)
    {
        for(int k=0;k<size-1;k++)
        {
            if(stringsort(word[k],word[k+1])==1)
            {
                char temp[100];             //sorting the string
                stringcopy(temp,word[k]);
                stringcopy(word[k],word[k+1]);
                stringcopy(word[k+1],temp);
            }
        }
    }
    for(i=0;i<size;i++)
    {
        printf("%s %d ",word[i],num[i]);
    }
```

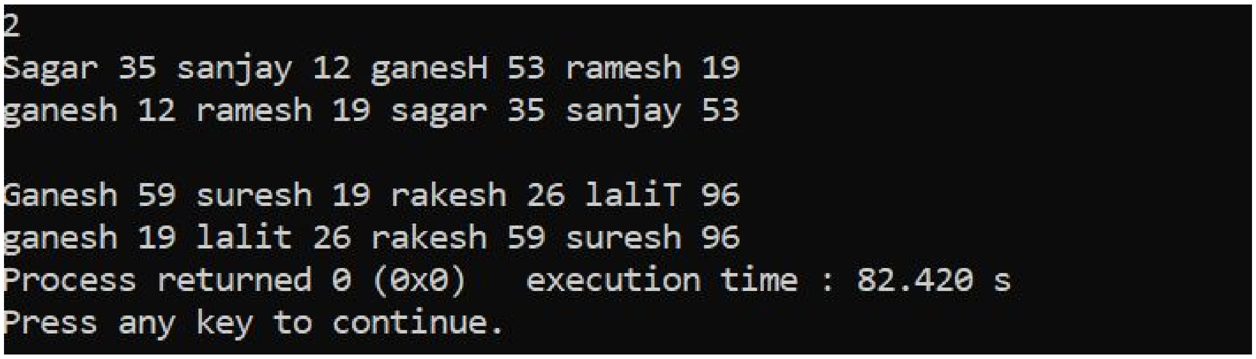
Fig 3.2.3



```
    }
    return 0;
}
```

Fig 3.2.4

3.3. Output:



```
2
Sagar 35 sanjay 12 ganesH 53 ramesh 19
ganesh 12 ramesh 19 sagar 35 sanjay 53

Ganesh 59 suresh 19 rakesh 26 laliT 96
ganesh 19 lalit 26 rakesh 59 suresh 96
Process returned 0 (0x0)   execution time : 82.420 s
Press any key to continue.
```

Fig 3.3.3

4. Problem 4
Prime Fibonacci

In this problem we need to form unique numbers using all the prime numbers in the given range and taking the count of the prime numbers in the list of unique numbers, maximum and minimum prime number as the first two elements of the Fibonacci series, find the last number of the Fibonacci series.

4.1. Problem Statement:

Given two numbers n1 and n2

- Find prime numbers between n1 and n2, then
- Make all possible unique combinations of numbers from the list of the prime numbers you found in step 1.
- From this new list, again find all prime numbers.
- Find the smallest (a) and largest (b) number from the 2nd generated list, also count of this list.
- Consider the smallest and largest number as the 1st and 2nd number to generate the Fibonacci series respectively till the count (number of primes in the 2nd list).
- Print the last number of a Fibonacci series as an output

Constraints

2 <= n1, n2 <= 100
n2 - n1 >= 35

Input Format

One line containing two space-separated integers n1 and n2.

Output

The last number of a generated Fibonacci series.

Timeout
1

Test Cases:

Example 1:

Input:
2 40
Output:
13158006689

Explanation:

1st prime list = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]

Combination of all the primes = [23, 25, 27, 211, 213, 217, 219, 223, 229, 231, 32, 35, 37, 311, 313, 319, 323, 329, 331, 337, 52, 53, 57, 511, 513, 517, 519, 523, 529, 531, 537, 72, 73, 75, 711, 713, 717, 719, 723, 729, 731, 737, 112, 113, 115, 117, 1113, 1117, 1119, 1123, 1129, 1131, 1137, 132, 133, 135, 137, 1311, 1317, 1319, 1323, 1329, 1331, 1337, 172, 173, 175, 177, 1711, 1713, 1719, 1723, 1729, 1731, 1737, 192, 193, 195, 197, 1911, 1913, 1917, 1923, 1929, 1931, 1937, 232, 233, 235, 237, 2311, 2313, 2317, 2319, 2329, 2331, 2337, 292, 293, 295, 297, 2911, 2913, 2917, 2919, 2923, 2931, 2937, 312, 315, 317, 3111, 3113, 3117, 3119, 3123, 3129, 3137, 372, 373, 375, 377, 3711, 3713, 3717, 3719, 3723, 3729, 3731]

2nd prime list=[193, 3137, 197, 2311, 3719, 73, 137, 331, 523, 1931, 719, 337, 211, 23, 1117, 223, 1123, 229, 37, 293, 2917, 1319, 1129, 233, 173, 3119, 113, 53, 373, 311, 313, 1913, 1723, 317]

smallest (a) = 23

largest (b) = 3719

Therefore, the last number of a Fibonacci series i.e. 34th Fibonacci number in the series that has 23 and 3719 as the first 2 numbers is 13158006689

Example 2:

Input:
30 70
Output:
2027041

Explanation:

1st prime list=[31, 37, 41, 43, 47, 53, 59, 61, 67]

2nd prime list generated form combination of 1st prime list = [3137, 5953, 5347, 6761, 3761, 4337, 6737, 6131, 3767, 4759, 4153, 3167, 4159, 6143]

the smallest prime in 2nd list=3137
the largest prime in 2nd list=6761

Therefore, the last number of a Fibonacci series i.e. 14th Fibonacci number in the series that has 3137 and 6761 as the first 2 numbers is 2027041

Concepts Used to Solve:

- **Function:** A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.

  Defining a function:

  ```
  return_type function_name( parameter list )
   {body of the function}
  ```

- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
  Syntax:

  The syntax of a for loop in C programming language is −

  ```
  for ( init; condition; increment)
  {
          statement(s);
  }
  ```

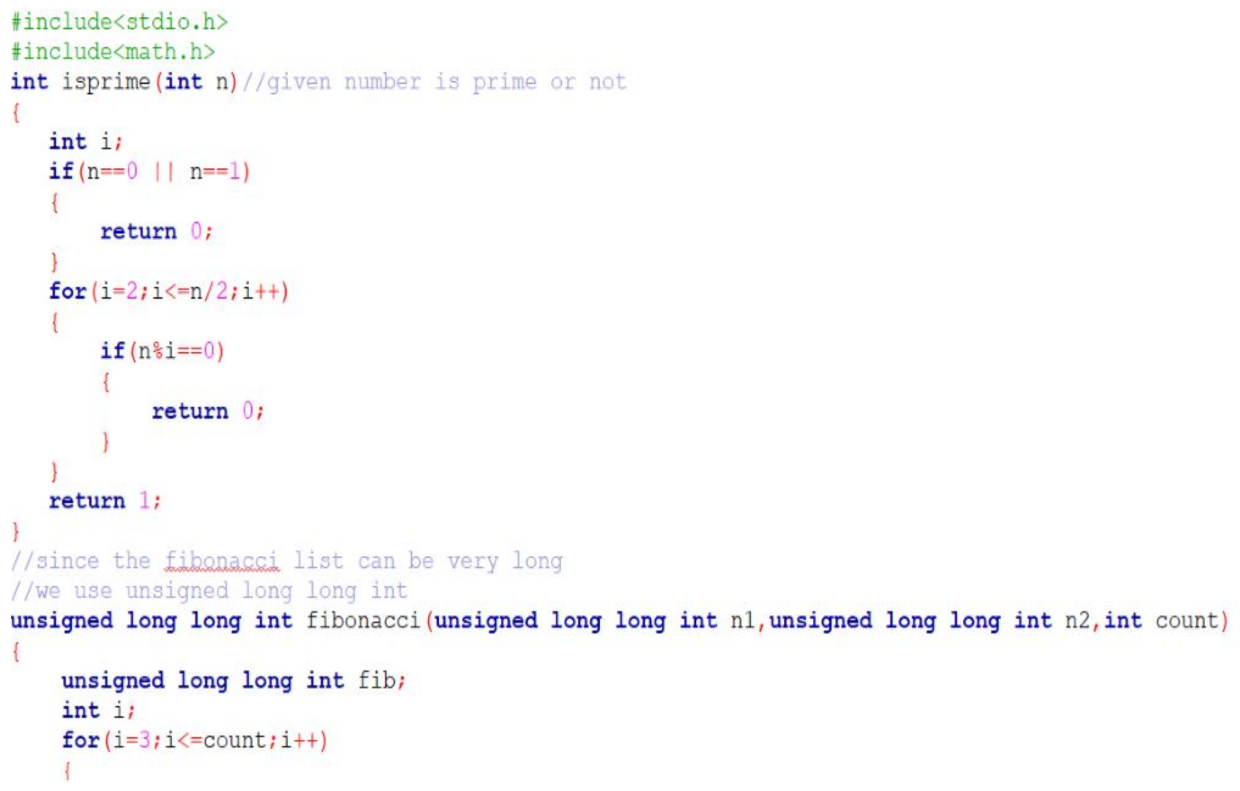
- **Switch Statement:** The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression. Switch is a control statement that allows a value to change control of execution.

- **Conditional Operator:** The conditional operator (? :) is a ternary operator (takes three operands). The conditional operator works as follows:

  The first operand is implicitly converted to bool. It is evaluated and all side effects are completed before continuing. If the first operand evaluates to true (1), the second operand is evaluated.

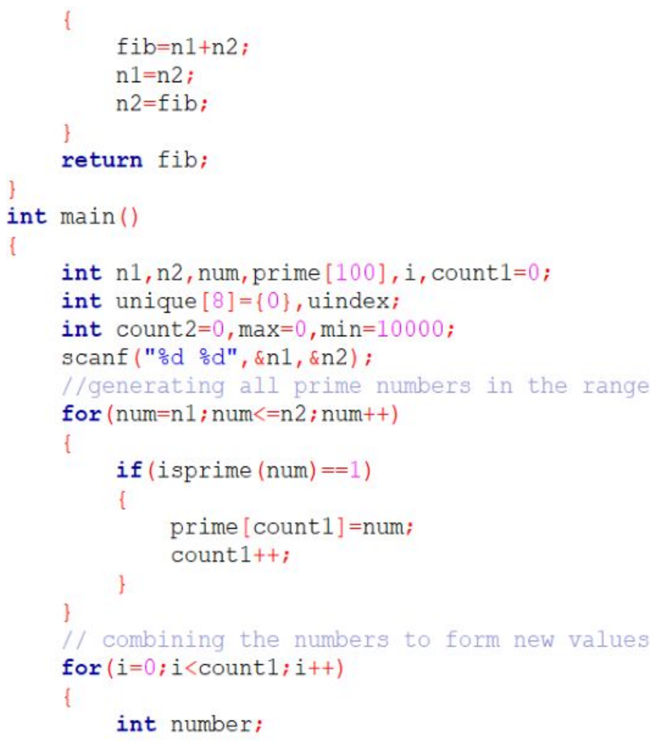
- **Arrays:** An array is a collection of data items, all of the same type, accessed using a common name. A one-dimensional array is like a list; A two dimensional array is like a table; The C language places no limits on the number of dimensions in an array, though specific implementations may.

4.2. Coding:



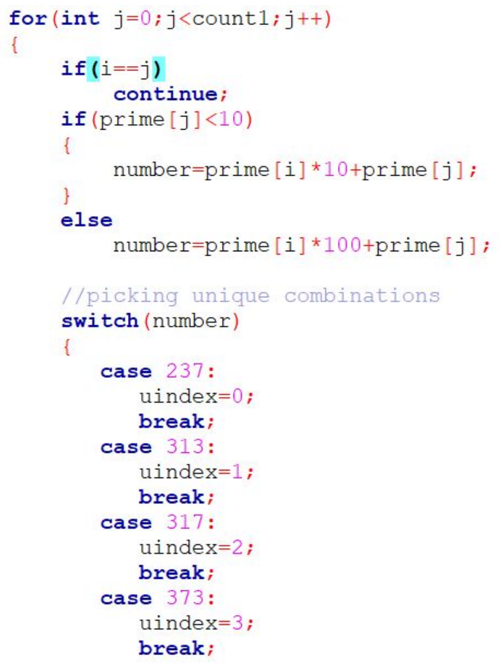
```c
#include<stdio.h>
#include<math.h>
int isprime(int n)//given number is prime or not
{
   int i;
   if(n==0 || n==1)
   {
       return 0;
   }
   for(i=2;i<=n/2;i++)
   {
       if(n%i==0)
       {
           return 0;
       }
   }
   return 1;
}
//since the fibonacci list can be very long
//we use unsigned long long int
unsigned long long int fibonacci(unsigned long long int n1,unsigned long long int n2,int count)
{
    unsigned long long int fib;
    int i;
    for(i=3;i<=count;i++)
    {
```

Fig 4.2.1



```
    {
        fib=n1+n2;
        n1=n2;
        n2=fib;
    }
    return fib;
}
int main()
{
    int n1,n2,num,prime[100],i,count1=0;
    int unique[8]={0},uindex;
    int count2=0,max=0,min=10000;
    scanf("%d %d",&n1,&n2);
    //generating all prime numbers in the range
    for(num=n1;num<=n2;num++)
    {
        if(isprime(num)==1)
        {
            prime[count1]=num;
            count1++;
        }
    }
    // combining the numbers to form new values
    for(i=0;i<count1;i++)
    {
        int number;
```

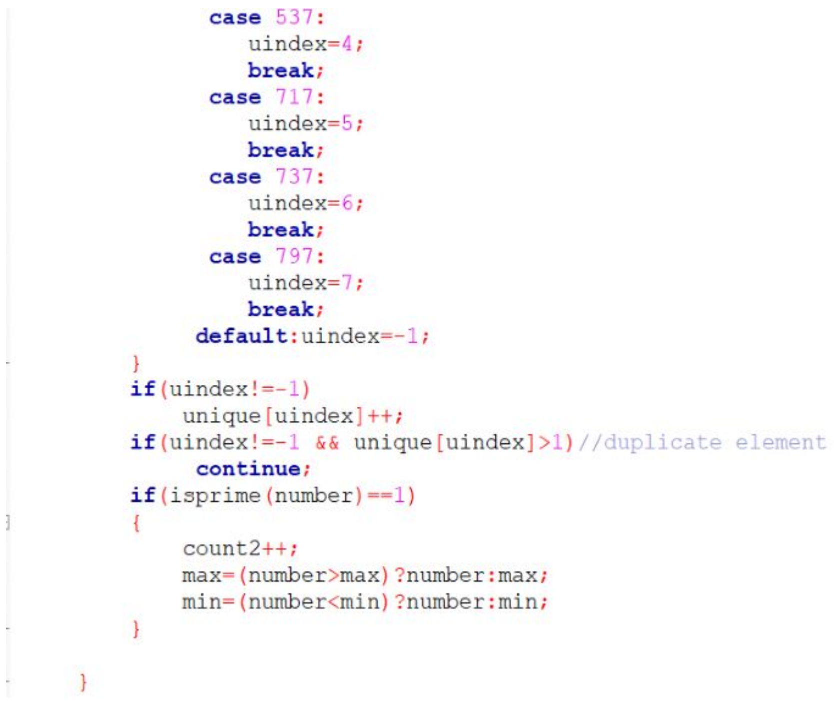
Fig 4.2.2



```
for(int j=0;j<count1;j++)
{
    if(i==j)
        continue;
    if(prime[j]<10)
    {
        number=prime[i]*10+prime[j];
    }
    else
        number=prime[i]*100+prime[j];

    //picking unique combinations
    switch(number)
    {
       case 237:
          uindex=0;
          break;
       case 313:
          uindex=1;
          break;
       case 317:
          uindex=2;
          break;
       case 373:
          uindex=3;
          break;
```

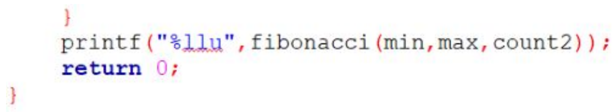
Fig 4.2.3



```
                case 537:
                   uindex=4;
                   break;
                case 717:
                   uindex=5;
                   break;
                case 737:
                   uindex=6;
                   break;
                case 797:
                   uindex=7;
                   break;
               default:uindex=-1;
          }
          if(uindex!=-1)
              unique[uindex]++;
          if(uindex!=-1 && unique[uindex]>1)//duplicate element
               continue;
          if(isprime(number)==1)
          {
              count2++;
              max=(number>max)?number:max;
              min=(number<min)?number:min;
          }

      }
```

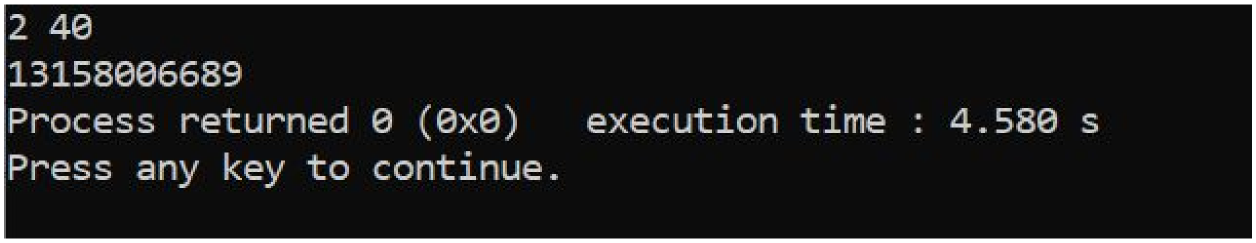
Fig 4.2.4



```
    }
    printf("%llu",fibonacci(min,max,count2));
    return 0;
}
```

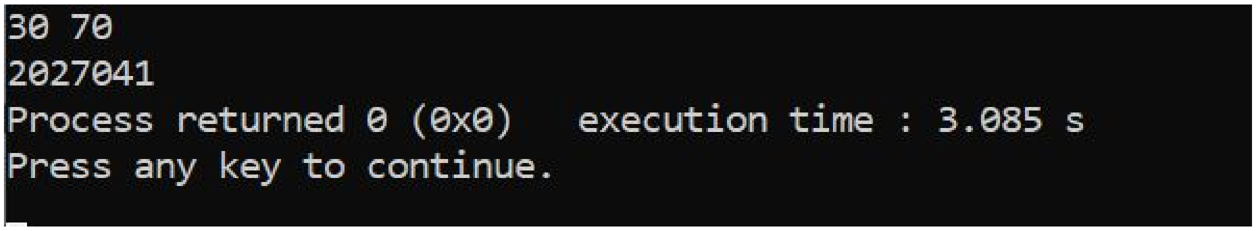
Fig 4.2.5

4.3. Output:



```
2 40
13158006689
Process returned 0 (0x0)   execution time : 4.580 s
Press any key to continue.
```

Fig 4.3.1



```
30 70
2027041
Process returned 0 (0x0)   execution time : 3.085 s
Press any key to continue.
```

Fig 4.3.2

5. Problem 4
Super ASCII String Checker

In this problem we give a new ASCII value to every alphabet like a=1, b=2, …, z=26 and check whether the count of each alphabet in the given string is equal to its new value.

5.1. Problem Statement:

In the Byteland country a string "S" is said to be a super ascii string if and only if count of each character in the string is equal to its ascii value.
In the Byteland country ascii code of 'a' is 1, 'b' is 2 ...'z' is 26.
Your task is to find out whether the given string is a super ascii string or not.

Input Format:

First line contains a number of test cases T, followed by T lines, each containing a string "S".

Output Format:

For each test case print "Yes" if the String "S" is super ascii, else print "No"

Constraints:

1<=T<=100
1<=|S|<=400, S will contain only lower case alphabets ('a'-'z').

Sample Input and Output:

SNo.	Input	Output
1	2 bba scca	Yes No

Explanation:
In case 1, viz. String "bba" -
The count of character 'b' is 2. Ascii value of 'b' is also 2.

The count of character 'a' is 1. Ascii value of 'a' is also 1.
Hence string "bba" is super ascii.

Concepts Used to Solve:

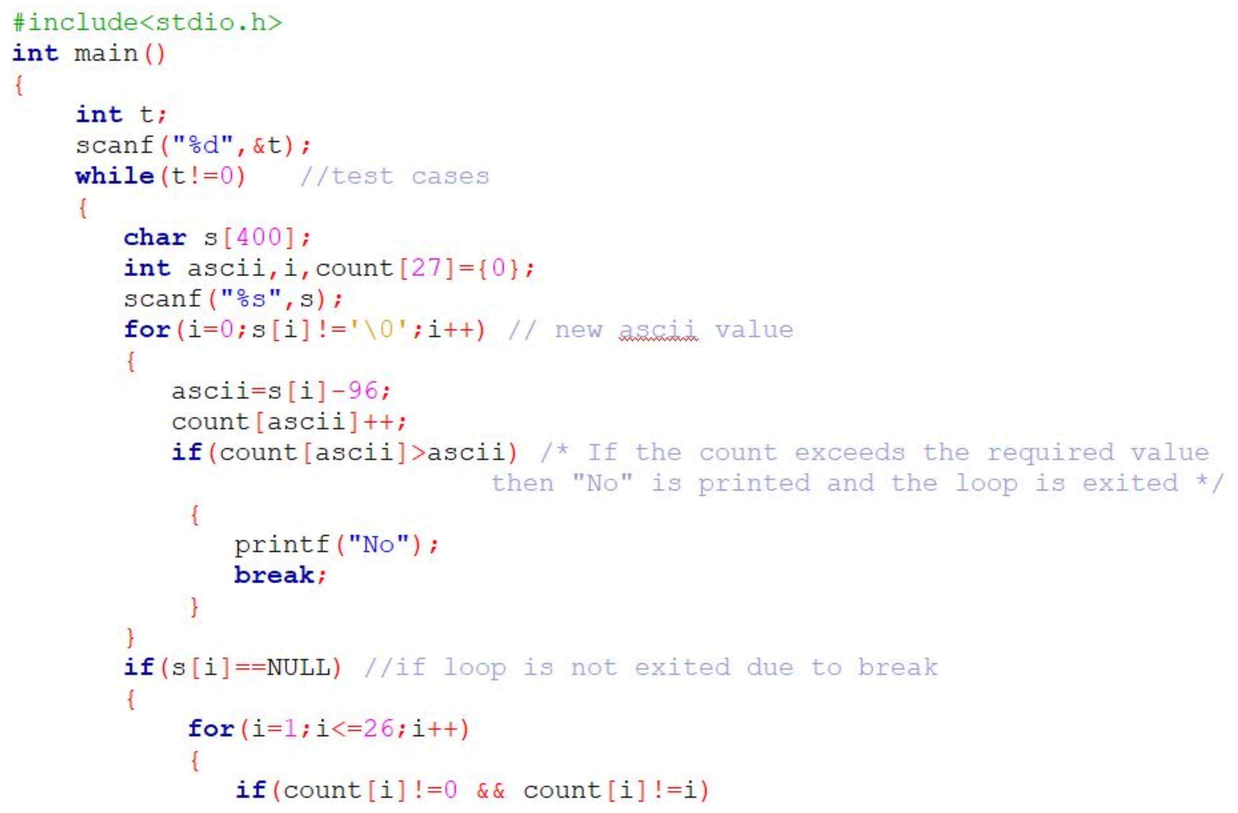
- **Array:** An array is a collection of data items, all of the same type, accessed using a common name. A one-dimensional array is like a list; A two dimensional array is like a table; The C language places no limits on the number of dimensions in an array, though specific implementations may.

- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
  Syntax:
  The syntax of a for loop in C programming language is −
  ```
  for ( init; condition; increment)
  {
          statement(s);
  }
  ```

- **While loop:** A while loop in C programming repeatedly executes a target statement as long as a given condition is true.
  Syntax :
  The syntax of a while loop in C programming language is −
  ```
  while(condition) {   statement(s);}
  ```

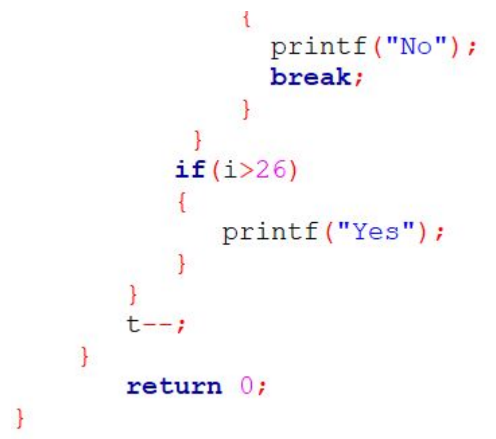
- **ASCII:** ASCII is the acronym for the American Standard Code for Information Interchange. It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127.

5.2. Coding:



```c
#include<stdio.h>
int main()
{
    int t;
    scanf("%d",&t);
    while(t!=0)   //test cases
    {
       char s[400];
       int ascii,i,count[27]={0};
       scanf("%s",s);
       for(i=0;s[i]!='\0';i++) // new ascii value
       {
          ascii=s[i]-96;
          count[ascii]++;
          if(count[ascii]>ascii) /* If the count exceeds the required value
                                    then "No" is printed and the loop is exited */
           {
              printf("No");
              break;
           }
       }
       if(s[i]==NULL) //if loop is not exited due to break
       {
           for(i=1;i<=26;i++)
           {
              if(count[i]!=0 && count[i]!=i)
```

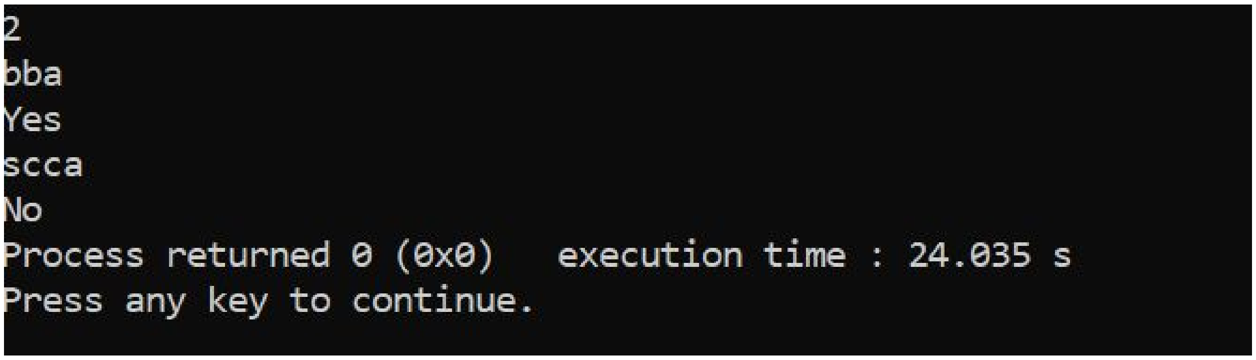
Fig 5.2.1



```c
                {
                  printf("No");
                  break;
                }
           }
          if(i>26)
          {
             printf("Yes");
          }
       }
       t--;
    }
       return 0;
}
```

Fig 5.2.2

5.3. Output:



```
2
bba
Yes
scca
No
Process returned 0 (0x0)   execution time : 24.035 s
Press any key to continue.
```

Fig 5.3.1

6. Problem 5
Consecutive Prime Sum

In this problem we need to find the count of sum of consecutive prime numbers such that the sum is also prime.

6.1. Problem Statement:

Some prime numbers can be expressed as a sum of other consecutive prime numbers.

For example,

5 = 2 + 3
17 = 2 + 3 + 5 + 7
41 = 2 + 3 + 5 + 7 + 11 + 13.
Your task is to find out how many prime numbers which satisfy this property are present in the range 3 to N subject to a constraint that summation should always start with number 2.
Write code to find out the number of prime numbers that satisfy the above-mentioned property in a given range.

Input Format:

First line contains a number N

Output Format:

Print the total number of all such prime numbers which are less than or equal to N.

Constraints:

2<N<=12,000,000,000

Concepts Used to Solve:

- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

The syntax of a for loop in C programming language is −

```
for ( init; condition; increment)
{
   statement(s);
}
```

- **If-else statement:** If the Boolean expression evaluates to true, then the if block will be executed, otherwise, the else block will be executed. C programming language assumes any non-zero and non-null values as true and if it is either zero or null, then it is assumed as false value.

```
if(boolean_expression)
       { /* statement(s) will execute if the boolean expression
   is true */}
   else
         { /*statement(s) will execute if the boolean expression
is false */}
```

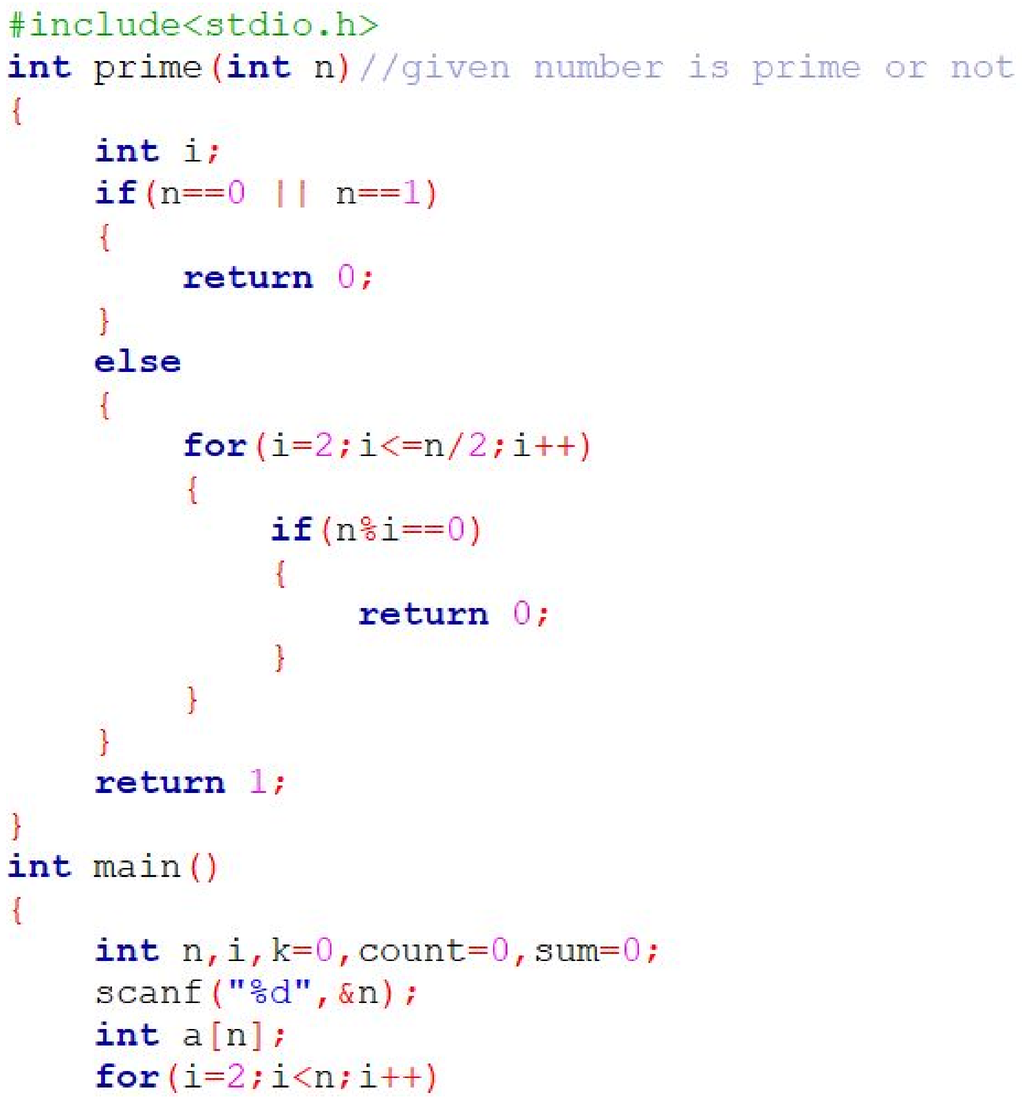
- **Arrays:** An array is a collection of data items, all of the same type, accessed using a common name. A one-dimensional array is like a list; A two dimensional array is like a table; The C language places no limits on the number of dimensions in an array, though specific implementations may.

- **Function:** A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.

Defining a function:

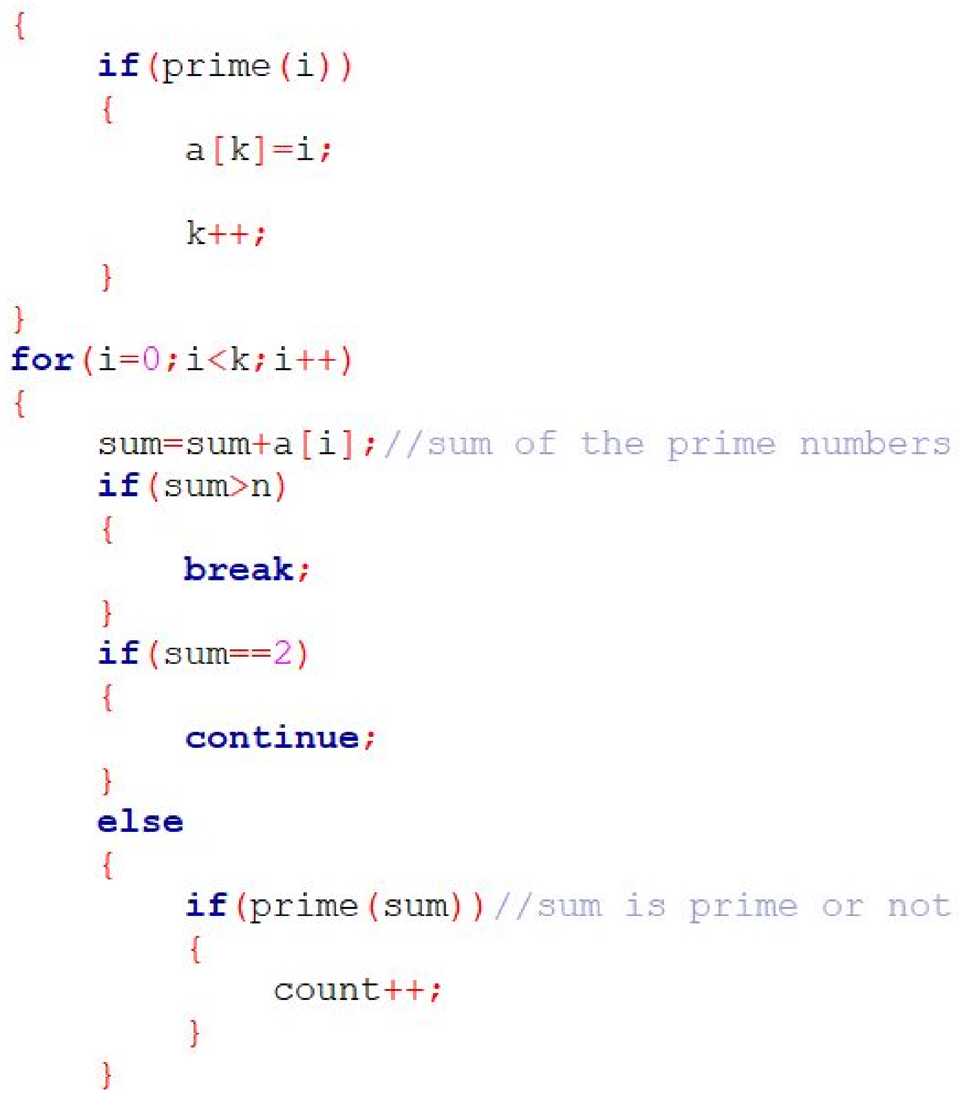
```
      return_type function_name( parameter list)   {body of the
   function}
```

6.2. Coding:



```c
#include<stdio.h>
int prime(int n)//given number is prime or not
{
    int i;
    if(n==0 || n==1)
    {
        return 0;
    }
    else
    {
        for(i=2;i<=n/2;i++)
        {
            if(n%i==0)
            {
                return 0;
            }
        }
    }
    return 1;
}
int main()
{
    int n,i,k=0,count=0,sum=0;
    scanf("%d",&n);
    int a[n];
    for(i=2;i<n;i++)
```

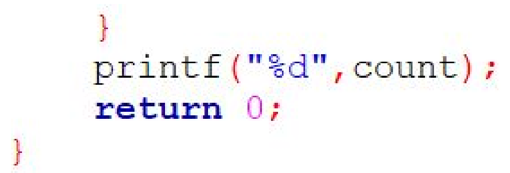
Fig 6.2.1



```c
{
    if(prime(i))
    {
        a[k]=i;

        k++;
    }
}
for(i=0;i<k;i++)
{
    sum=sum+a[i];//sum of the prime numbers
    if(sum>n)
    {
        break;
    }
    if(sum==2)
    {
        continue;
    }
    else
    {
        if(prime(sum))//sum is prime or not
        {
            count++;
        }
    }
```

Fig 6.2.2



```c
    }
    printf("%d",count);
    return 0;
}
```

Fig 6.2.3

6.3. Output:

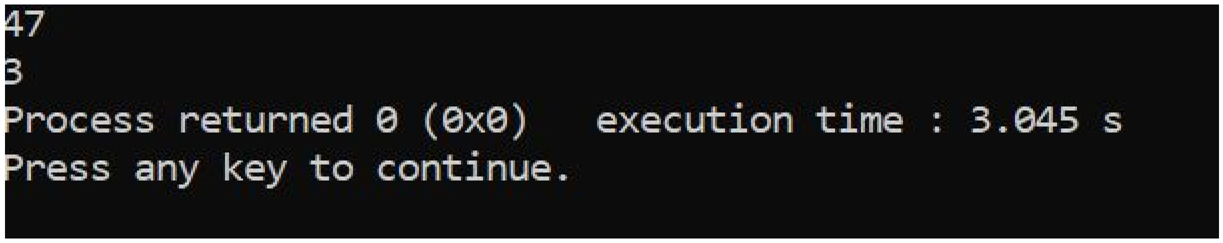

```
47
3
Process returned 0 (0x0)   execution time : 3.045 s
Press any key to continue.
```

Fig 6.3.1

7. Problem 6
Houses Problem

In this problem we need to find the maximum value stolen by the thief.

7.1. Problem Statement:

There are n houses built in a line, each of which contains some value in it.
A thief is going to steal the maximal value of these houses, but he can't steal in two adjacent houses because the owner of the stolen houses will tell his two neighbors left and right side.
What is the maximum stolen value?

Sample Input:

7
6, 7, 1, 3, 8, 2, 5

Sample Output:

20

Concepts Used to Solve:

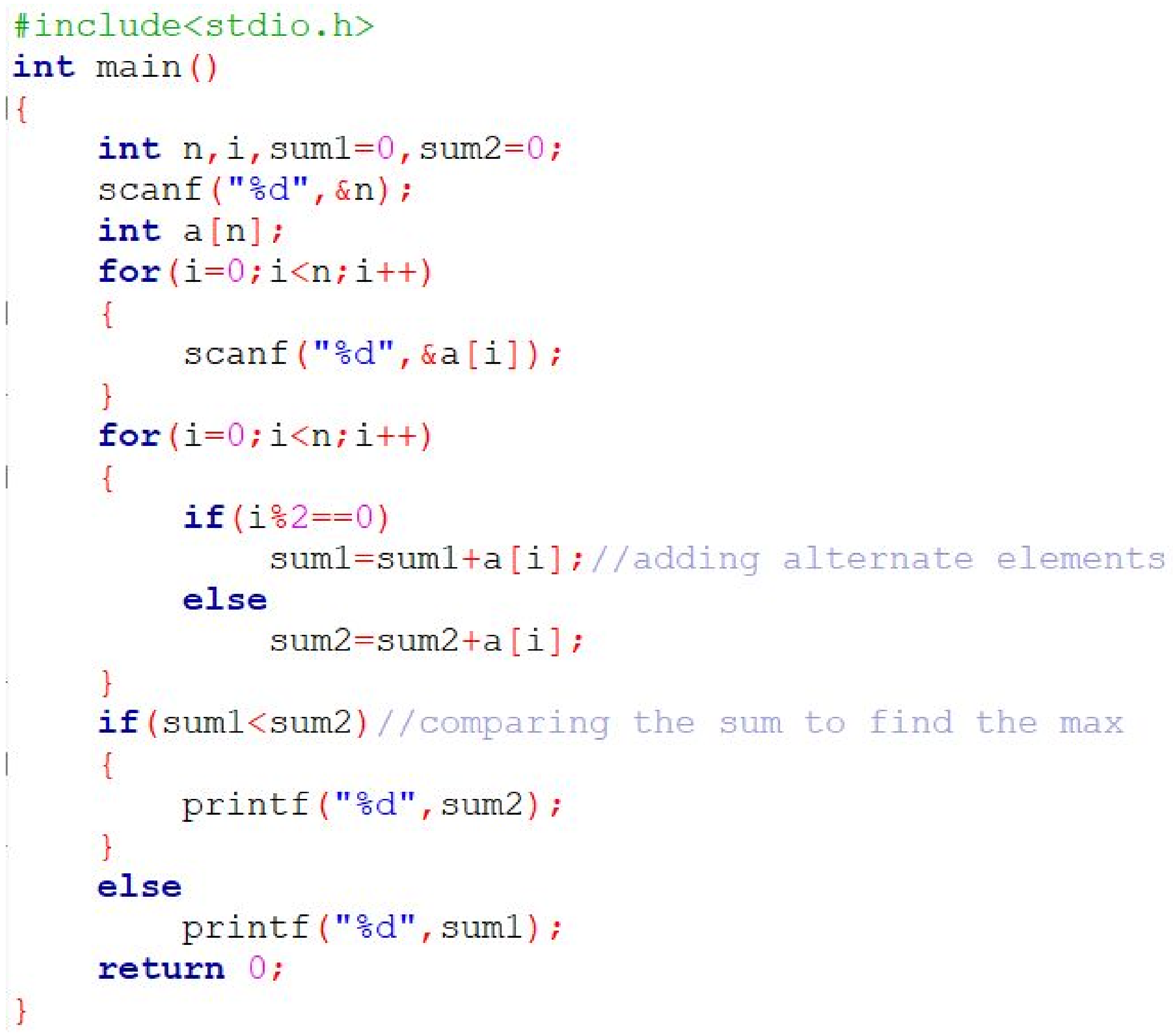
- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
  Syntax:
  The syntax of a for loop in C programming language is −
  ```
   for ( init; condition; increment)
  {
     statement(s);
  }
  ```

- **If-else statement:** If the Boolean expression evaluates to true, then the if block will be executed, otherwise, the else block will be executed. C programming language assumes any non-zero and non-null values as true and if it is either zero or null, then it is assumed as false value.

```
 if(boolean_expression)
       { /* statement(s) will execute if the boolean expression
   is true */}
else
       { /*statement(s) will execute if the boolean expression
is false */}
```

- **Arrays:** An array is a collection of data items, all of the same type, accessed using a common name. A one-dimensional array is like a list; A two dimensional array is like a table; The C language places no limits on the number of dimensions in an array, though specific implementations may.

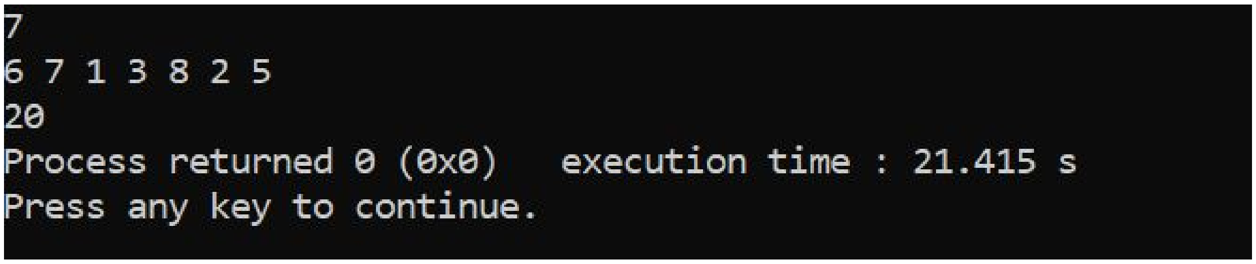
7.2. Coding:



```
#include<stdio.h>
int main()
{
    int n,i,sum1=0,sum2=0;
    scanf("%d",&n);
    int a[n];
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n;i++)
    {
        if(i%2==0)
            sum1=sum1+a[i];//adding alternate elements
        else
            sum2=sum2+a[i];
    }
    if(sum1<sum2)//comparing the sum to find the max
    {
        printf("%d",sum2);
    }
    else
        printf("%d",sum1);
    return 0;
}
```

Fig 7.2.1

7.3. Output:



```
7
6 7 1 3 8 2 5
20
Process returned 0 (0x0)   execution time : 21.415 s
Press any key to continue.
```

Fig 7.3.1

8. Software Requirements

8.1. Hardware Requirements:

This project can be executed in any system or an android phone without prior to any platform. We can use any online compiler and interpreter.

8.2. Software Requirements:

There are two ways to execute these projects

1. Online compilers
2. Software for execution (DEV C++, CodeBlocks, ….)

Online Compilers require only internet connection. We have many free compilers with which we can code.
Softwares for execution need to be installed based on the user's system specification. These help us to completely execute the project. These softwares are based on the platforms.

BIBLIOGRAPHY

- https://www.programminggeek.in/2013/08/solution-of-zombie-world-problem-of-round1-of-codevita-2013-organized-by-TCS.html#.Xx102p4zY2w

- https://www.programminggeek.in/2015/08/tcs-codevita-2015-round-2-alpha-numeric-sorting.html#.Xx11N54zY2w

- https://www.programmersdoor.com/post/tcs-codevita-prime-fibonacci

- https://www.programminggeek.in/2014/09/tcs-codevita-2014-questions-round1-set1.html#.Xx11o54zY2w

- https://www.programminggeek.in/2016/07/tcs-codevita-2016-round1-question.html#.Xx113p4zY2w

- https://prepinsta.com/tcs-codevita/python-program-for-houses-problem/