Godavari College Of Engineering, Jalgaon.

DEPARTMENT OF

COMPUTER ENGINEERING

V SEMESTER

LAB MANUAL

Competitive Programming-I

SUBJECT CODE: BTCOC506

SESSION : Auguest 2020 – December 2020

Faculty : Prof. Swapnil Shete.

Index

Unite Number	Practical Number	Practicals Name	Dates
1	1	The 3n + 1 Problem	09-09-2020
	2	Minesweeper	09-09-2020
	3	Interpreter	18-09-2020
2	4	Jolly	18-09-2020
	5	Poker Hands	01-10-2020
	6	Crypt Kicker	01-10-2020
	7	Contest Scoreboard	08-10-2020
3	8	WERTYU	08-10-2020
	9	Common Permutation	19-10-2020
	10	Automated Judge Script	19-10-2020
	11	File Fragmentation	30-10-2020
4	12	Vito's Family	30-10-2020
	13	Stacks of Flapjacks	08-11-2020
	14	Bridge	08-11-2020
	15	Shell Sort	23-11-2020
5	16	Primary Arithmetic	23-11-2020
	17	Reverse and Add	28-11-2020
	18	A Multiplication Game	28-11-2020
	19	Polynomial Coefficients	02-12-2020
	20	How Many Fibs?	02-12-2020
	21	Counting	04-12-2020

6	22	Expressions	04-12-2020
	23	Self-describing Sequence	04-12-2020

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 1	**Date: 09-09-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on 3n+1 Problem.

Aim: Implement the 3n+1 program.

Theory:

Consider the following algorithm to generate a sequence of numbers. Start with an integer n. If n is even, divide by 2. If n is odd, multiply by 3 and add 1. Repeat this process with the new value of n, terminating when n = 1.

For example, the following sequence of numbers will be generated for n = 22:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured (but not yet proven) that this algorithm will terminate at n = 1 for every integer n. Still, the conjecture holds for all integers up to at least 1,000,000. For an input n, the cycle-length of n is the number of numbers generated up to and including the 1.

In the example above, the cycle length of 22 is 16.

Given an integer i determine the cycle length .

Input Format : An integer n

Output Format : An integer : cycle length

Sample Input : 22

Sample Output : 16

Explanation

for n = 22:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Cycle length = 16

Source Code:

```
#include<stdio.h>
int main ()
{
        int n, m,i,k,j,c,s;
```

```
        while  (scanf("%d %d",&n,&m)==2)
        {
                s=0;
                printf ("%d %d", n,m);
                if(n>m)
                {
                        k=m;
                        m=n;
                        n=k;
                }
                for(i=n;i<=m; i++)
                {
                        c=1;
                        j=i;
                        while(j > 1)
                {
                        if(j % 2==0)
                        j=j/2;
                        else
                        j=(3*j)+1;
                        c++;
                }
                if(c>=s)
                s=c;
        }
        printf ("%d",s);
    }
    return 0;
}
```

Output:

Conclusion: In this Practical I learn to implement the 3n+1 program.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 2	**Date: 09-09-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Program on Minesweeper.

Aim: To implement the minesweeper Program.

Theory:

Minesweeper is a single- player puzzle computer game. The objective of the game is to clear a rectangular board containing hidden "mines" or bombs without detonating any of them, with help from clues about the number of neighboring mines in each field.

Rule:

1. The board is a two-dimensional space, which has a predetermined number of mines.
2. Cells have two states, opened and closed.
3. If you left-click on a closed cell:
    1. Cell is empty and opened.
        1. If neighbor cell(s) have mine(s), this opened cell shows neighbor mine count.
        2. If neighbor cells have no mines, all neighbor cells are opened automatically.
    2. Cell has a mine, game ends with FAIL.
4. If you right-click on a closed cell, you put a flag which shows that "I know this cell has a mine".
5. If you multi-click (both right and left click) on a cell which is opened and has at least one mine on its neighbors:
    1. If neighbor cells' total flag count equals to this multi-clicked cell's count and predicted mine locations are true, all closed and unflagged neighbor cells are opened automatically.
    2. If neighbor cells' total flag count equals to this multi-clicked cell's count and at least one predicted mine location is wrong, game ends with FAIL.
6. If all cells (without mines) are opened using left clicks and/or multi-clicks, game ends with success.

Source Code:

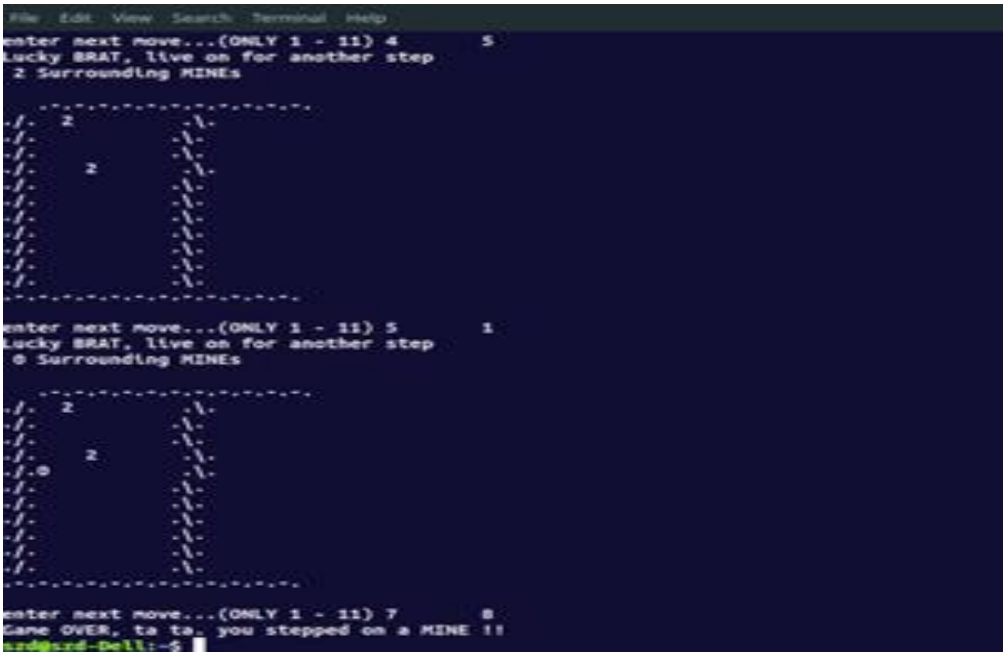
```
#include<stdio.h>
```

```
#include<stdlib.h>

void welcome();
void rand_mines(char msweep[12][12]);
void printmatrix(char msweep[12][12],int r,char user_chart[12][12]);
 int process(char msweep[12][12],int r,int c,charuser_chart[12][12]);

int main()

{

        msweep[12][12] = {{'0'}};
       int i,r,c;
       char user_chart[12][12] = {{'0'}};

        // welcome();

        rand_mines(mswep;

      // printmatrix(msweep,12,user_chart);              // note grid from 1 to 11

        printf("Enter your location(ONLY 1 - 11) on the minefield x,y\n");
        scanf("%d%d",&r,&c);

        printmatrix(msweep,12,user_chart);

        i = process(msweep,r,c,user_chart); //returns 1 or 0,1 is notmine 0 =mine


        while(i == 1)
        {
                printf("Lucky BRAT, live on for another step\n"); printf("
                %c Surrounding MINEs\n\n",msweep[r][c]);

                printmatrix(msweep,12,user_chart);

                printf("enter next move...(ONLY 1 - 11) ");
                scanf("%d%d",&r,&c);
                i=0


                i = process(msweep,r,c,user_chart);


        }

        if(i==0)
        printf("Game OVER, ta ta. you stepped on a MINE !!\n"); return 0;

}

void welcome()
{

        char op; // opereation
    printf("Welcome to MINESWEEPER in C >>.  \n");

        printf("Enter <<\n");
        printf("               i for  instructions\n"); printf("   any other
        key to enter game\n"); scanf("%c",&op);

        if(op == 'i')
        {
```

```
            printf("OH DEAR, what a shock you are unfortunatly in the midst of a "); printf("mine
    field.\n");
            printf("Enter the coordinates of the x and y plane between 1 to 11\n"); printf("Are you
    destined to DIE or live ?\n");
            printf("HA ha ha hah, GOOD LUCK\n\n");

        }
        else
        returrn;
}

void rand_mines(char msweep[12][12])

{

  int r,c,m;

  //srand(12);

  for(m=0;m<20;m++) // plant 20 rand mines(m

    {
        r = rand() % 13; // this is mine planting
                                // so can be at the edges aswell c = rand()
        % 13; // so 0 to 13 is APPROPRIATE.


        msweep[r][c] = '9';
      printf("%d %d \n",r,c);

    }

return;

}

void printmatrix(char msweep[][12],int r,char user_chart[12][12])


{

int i,j;

printf("            .-.-.-.-.-.-.-.-.-.-.-.\n");

  for(i=1;i<r;i++)
    {
      printf("./.");

      for(j=1;j<12;j++) //printing 1 to 11
        {
          printf("%c ",user_chart[i][j]);//to refer to mines use msweep[i][j]
        }

      printf(".\\.");

      printf("\n");

    }

printf(".-.-.-.-.-.-.-.-.-.-.-.-.\n\n");
```

```
 return;
 }

 int process(char msweep[12][12],int r,int c,char user_chart[12][12])
 {

  int i=r,j=c,b=0,k; char
  C;

  if(msweep[r][c] == '9')
  {     k=0;
      return k;
  }
  else
    {
      if(msweep[i-1][j-1] == '9') b++;
      if(msweep[i-1][j] == '9') b++;
      if(msweep[i-1][j+1] == '9') b++;
      if(msweep[i][j-1] == '9') b++;
      if(msweep[i][j+1] == '9') b++;
      if(msweep[i+1][j-1] == '9') b++;
      if(msweep[i+1][j] == '9') b++;
      if(msweep[i+1][j+1] == '9') b++;

    C = (char)(((int)'0')+b); // to covert int to char; msweep[r][c] = C;
      user_chart[r][c] = C;

    }

  return 1;

}
```

Output:



Conclusion: In this Practical I learn to implement the minesweeper Program.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 3	**Date: 18-09-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on Interpreter.

Aim: Implement the Interpreter program.

Theory:

An interpreter is a computer program that is used to directly execute program instructions written using one of the many high-level programming languages. The interpreter transforms the high-level program into an intermediate language that it then executes, or it could parse the high-level source code and then performs the commands directly, which is done line by line or statement by statement.

An **Interpreter** directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code. Examples of interpreted languages are Perl, Python and Matlab.

Source Code:

```
INTEGER, PLUS, EOF = 'INTEGER', 'PLUS', 'EOF'

class Token(object):
    def __init__(self, type, value):
        self.type = type
        self.value = value

    def __str__(self):
        return 'Token({type}, {value})'.format(
            type=self.type,
            value=repr(self.value)
        )
    def __repr__(self):
        return self.__str__()
class Interpreter(object):

    def __init__(self, text):
        self.text = text
        self.pos = 0
        self.current_token = None
    def error(self):
        raise Exception('Error parsing input')

    def get_next_token(self):
        text = self.text
        if self.pos > len(text) - 1:
            return Token(EOF, None)
```

```
        current_char = text[self.pos]
        if current_char.isdigit():
            token = Token(INTEGER, int(current_char))
            self.pos += 1
            return token
        if current_char == '+':
            token = Token(PLUS, current_char)
            self.pos += 1
            return token
        self.error()

    def eat(self, token_type):
        if self.current_token.type == token_type:
            self.current_token = self.get_next_token()
        else:
            self.error()

    def expr(self):
        self.current_token = self.get_next_token()
        left = self.current_token
        op = self.current_token
        self.eat(PLUS)
        right = self.current_token
 result = left.value + right.value
        return result


def main():
    while True:
        try:
            text = input('>>> ')
        except EOFError:
            break
        if not text:
            continue
        interpreter = Interpreter(text)
        result = interpreter.expr()
        print(result)


if __name__ == '__main__':
    main()
```

Output:

Conclusion: In this Practical I learn to implement the Interpreter program.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 4	**Date: 18-09-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on jolly jumper.

Aim: Implement program on jolly jumper.

Theory:

A sequence of n>0 integer is called a jolly jumper if the absolute values of the difference between successive elements take on all the values 1 through n-1 . for instance, 1 4 2 3

Is a jolly jumper, because the absolute difference are 3, 2, and 1 respectively? The definition implies that any sequence of a single integer is a jolly jumper. You are to write a program to determine whether or not each of a number of sequence is a jolly.

Input : Each line of input contain an integer n <= 3000 followed by n integers representing the sequence.

Output : For each line of input, generate a line of output saying “Jolly” or “Not jolly”

Sample Input : 4 1 4 2 3

5 1 4 2 -1 6

Sample Output : JOLLY

NOT JOLLY

Program :

```
#include <stdio.h>

#define MAX 3000

int main(){

static int N, I, J, V[MAX], A[MAX];

while(scanf("%d",&N) == 1){

for(I = 0; I < N; I++){

scanf("%d",&V[I]);

A[I] = 0;

}
```

```
J = N-1;
for(I = 0; I < J; I++)
A[abs(V[I]-V[I+1])] = 1;
J = 1;
for(I = 1; I < N; I++){
if(!A[I]){
J = 0;
break;
}
}
if(J)
printf("Jolly\n");
else
printf("Not jolly\n");
}
return 0;
}
```

Output:

Conclusion : In this Practical I learn to Implement program on jolly jumper.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 5	**Date: 01-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on Poker Hands.

Aim: Implement program on poker hands.

Theory:

Stud Poker

The stud poker takes place with a deck of 52 cards. Every card has a value and a color (Spades, Hearts, Clubs, Diamonds). The values are ordered as follows from the weaker to the stronger: from 2 to 10, then Jack, Queen, King, Ace (which is also 1).

In the stud poker, every player has five cards in hand. Five cards of the hand of a player form a combination. Here are the various combinations, classified in increasing order by value.

1. **High card:** None of the following combinations:

2. **One pair:** two cards of the same value.

3. **Two pairs**: two times two cards of same value.

4. **Three of a kind**: Three cards of the same value.

5. **Straight:** five cards with values in sequence (Ace can also be seen as one, but we must choose: The values "Ace-2-3-4-5" form a sequence, like the values "10 - Jack - Queen - King - Ace ", but, for example, values" King - Ace -2 - 3 - 4 "are not a sequence).

6. **Flush(color)**: Five cards of same color.

7. **Full house:** three of a kind and one pair.

8. **Four of a kind**: four cards of the same value.

9. **Straight Flush:** Straight in which the cards have the same color.

Work : A card from a deck of 52 cards will be represented by:

- An int between 1 and 13 for the card value (= 1 Ace, 11 = Jack, 12 = lQueen and 13 = King) and

- A char for the color of the card 'P' Spades, 'C' to Hearts 'T' for Clubs and 'K' for Diamonds.

Program:

```
#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

#define FACES "23456789tjqka"

#define SUITS "shdc"

typedef int bool;

typedef struct {

} card;

int face; /* FACES map to 0..12 respectively */

char suit;

card cards[5];

int compare_card(const void *a, const void *b)

{

card c1 = *(card *)a;

card c2 = *(card *)b;

return c1.face - c2.face;

}

bool equals_card(card c1, card c2)

{

if (c1.face == c2.face && c1.suit == c2.suit) return TRUE;}

return FALSE;

bool are_distinct()

{

int i, j;

for (i = 0; i < 4; ++i)
```

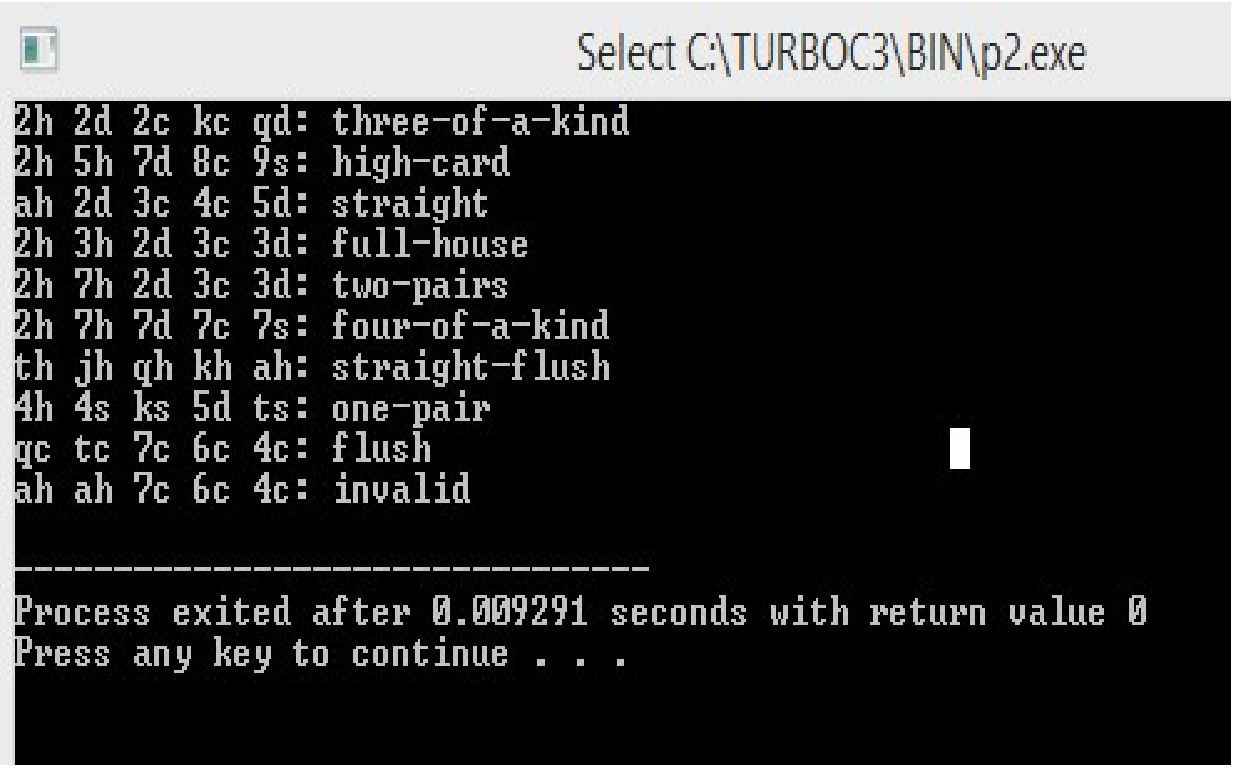
```
for (j = i + 1; j < 5; ++j)
if (equals_card(cards[i], cards[j])) return FALSE;
return TRUE;
}
bool is_straight()
{
int i;
qsort(cards, 5, sizeof(card), compare_card);
if (cards[0].face + 4 == cards[4].face) return TRUE;
if (cards[4].face == 12 && cards[0].face == 0 &&
cards[3].face == 3) return TRUE;
return FALSE;
}
bool is_flush()
{
int i;
char suit = cards[0].suit;
for (i = 1; i < 5; ++i) if (cards[i].suit != suit) return FALSE;
return TRUE;
}
const char *analyze_hand(const char *hand)
{
int i, j, gs = 0;
char suit, *cp;
bool found, flush, straight;
int groups[13];
if (strlen(hand) != 14) return "invalid";
for (i = 0; i < 14; i += 3)
{cp = strchr(FACES, tolower(hand[i]));
if (cp == NULL) return "invalid";
```

```
j = i / 3;
cards[j].face = cp - FACES;
suit = tolower(hand[i + 1]);
cp = strchr(SUITS, suit);
if (cp == NULL) return "invalid";
cards[j].suit = suit;
}
if (!are_distinct()) return "invalid";
for (i = 0; i < 13; ++i) groups[i] = 0;
for (i = 0; i < 5; ++i) groups[cards[i].face]++;
for (i = 0; i < 13; ++i) if (groups[i] > 0) gs++;
switch(gs)
{
case 2:
found = FALSE;
for (i = 0; i < 13; ++i) if (groups[i] == 4) {
found = TRUE;
break;
}
if (found) return "four-of-a-kind";
return "full-house";
case 3:
found = FALSE;
for (i = 0; i < 13; ++i) if (groups[i] == 3)
{
found = TRUE;
break;
}
if (found) return "three-of-a-kind";
return "two-pairs";
```

```
case 4:
return "one-pair";
default:
flush = is_flush();
straight = is_straight();
if (flush && straight)
return "straight-flush";}
else if (flush)
return "flush";
else if (straight)
return "straight";
else
return "high-card";
}
int main()
{
int i;
const char *type;
const char *hands[10] = {
"2h 2d 2c kc qd",
"2h 5h 7d 8c 9s",
"ah 2d 3c 4c 5d",
"2h 3h 2d 3c 3d",
"2h 7h 2d 3c 3d",
"2h 7h 7d 7c 7s",
"th jh qh kh ah",
"4h 4s ks 5d ts",
"qc tc 7c 6c 4c",
"ah ah 7c 6c 4c"
};
```

```
for (i = 0; i < 10; ++i) {

type = analyze_hand(hands[i]);

printf("%s: %s\n", hands[i], type);

}

return 0;

}
```

Output:



Conclusion : In this Practical I learn to Implement program on poker hands.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 6	**Date: 01-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a Program for Crypt Kicker.

Aim: Study and implementation of Crypt Kicker.

Theory:

A common but insecure method of encrypting text is to permute the letters of the alphabet. In other words, each letter of the alphabet is consistently replaced in the text by some other letter. To ensure that the encryption is reversible, no two letters are replaced by the same letter. Your task is to decrypt several encoded lines of text, assuming that each line uses a different set of replacements, and that all words in the decrypted text are from a dictionary of known words.

Input:

The input consists of a line containing an integer n, followed by n lowercase words, one per line, in alphabetical order. These n words compose the dictionary of words which may appear in the decrypted text. Following the dictionary are several lines of input. Each line is encrypted as described above. There are no more than 1,000 words in the dictionary. No word exceeds 16 letters. The encrypted lines contain only lower case letters and spaces and do not exceed 80 characters in length.

Output:

Decrypt each line and print it to standard output. If there are multiple solutions, any one will do. If there is no solution, replace every letter of the alphabet by an asterisk.

Source Code:

```
#include <cstdio>

#include <iostream>

#include <vector>

#include <string>

#include <stack>

using namespace std;// dicionario

vector<string> dicio[20][20]; // numero de letras, letras repetidas

vector<string> answer;
```

```
// tabela de traducao
char trans[30];
inline void set_translation(char src, char dest) { trans[src-'a']=dest; }
inline char get_translation(char src) { return trans[src-'a']; }
inline bool nothing_translates_to(char c) {
int i;
for(i=0; i<30; i++) {
if(trans[i] == c)
return false;
}
return true;
}
// repeated_letters()
// devolve o numero de letras repetidas
inline int repeated_letters(string str)
{
int i, cnt[30], sum = 0;
for(i=0; i<30; i++)
cnt[i] = 0;
for(i=0; i<(int)str.length(); i++)
cnt[ int(str[i]-'a') ]++;
for(i=0; i<30; i++)
sum += cnt[i];
}
return sum;
// backtrack()
// Backtracking - retorna true se achou solucao
bool backtrack(char *line, int deep)
{int i, j, n, rep;
char cword[20];
```

```
string word;
// volta
if(*line == '\0')
return true;
sscanf(line, "%s%n", cword, &n);
word = cword;
rep = repeated_letters(word);
// enumerando candidatos
for(i=0; i<(int)dicio[ word.length() ][rep].size(); i++) {
stack<char> st;
string decoded;
string candidate = dicio[word.length()][rep][i]; // candidato a ser traducao de
word
for(j=0; j<(int)word.length(); j++) {
if(get_translation(word[j]) == 0) {
if(nothing_translates_to(candidate[j])) {
set_translation(word[j], candidate[j]);
st.push(word[j]);
}
else
break;
}
decoded += get_translation(word[j]);
}
// candidato valido. prossiga!
if(decoded.length() == candidate.length() && decoded == candidate) {
answer.push_back(decoded);
if(backtrack(line+n, deep+1))
return true;
answer.pop_back();
```

```
}while(!st.empty()) {
set_translation(st.top(), 0);
st.pop();
}
}
return false;
}
// impossible()
// retorna true se estiver na cara que nao tem resposta
bool impossible(char *str)
{     char *p;
int n, i;
char word[100];
bool imp = true;
for(p=str; sscanf(p, "%s%n", word, &n)>0 && imp; p+=n) {
for(i=0; i<20 && imp; i++)
imp = dicio[string(word).length()][i].empty();
} return imp; }
// main()
// funcao principal
int main()
{ int i, n;
char c;
string line;
// preenche dicionario
getline(cin, line);
sscanf(line.c_str(), "%d", &n);
while(n--) {
getline(cin, line);
dicio[line.length()][repeated_letters(line)].push_back(line);
```

```
}// frases
while(getline(cin, line)) {
// inicio
answer.clear();
for(c='a'; c<='z'; c++)
set_translation(c, 0);
// backtracking
if(impossible((char*)line.c_str()) || !backtrack((char*)line.c_str(), 0)) {
for(i=0; i<(int)line.length(); i++)
putchar((line[i]>='a'&&line[i]<='z') ? '*' : ' '); }
else if(answer.size() > 0){
printf("%s", answer[0].c_str());
for(i=1; i<(int)answer.size(); i++)
printf(" %s", answer[i].c_str()); }
putchar('\n');
}} return 0; }
```

Output : 6

And

Dick

Jane

Puff

Spot

Yertle

bjvg xsb hxsn xsb qymm xsb rqat xsb pnetfn

xxxx yyy zzzz www yyyy aaa bbbb ccc dddddd

dick and jane and puff and spot and yertle

**** *** **** *** **** *** **** *** ******

Conclusion : In this Practical I learn to implementation of Crypt Kicker.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 7	**Date: 08-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a Program for contest Scoreboard.

Aim: Implementation of contest Scoreboard.

Theory:

Think the contest score boards are wrong? Here's your chance to come up with the right rankings. Contestants are ranked first by the number of problems solved (the more the better), then by decreasing amounts of penalty time. If two or more contestants are tied in both problems solved and penalty time, they are displayed in order of increasing team numbers. A problem is considered solved by a contestant if any of the submissions for that problem was judged correct. Penalty time is computed as the number of minutes it took for the first correct submission for a problem to be received plus 20 minutes for each incorrect submission received prior to the correct solution. Unsolved problems incur no time penalties.

Input:

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. Input consists of a snapshot of the judging queue, containing entries from some or all of contestants 1 through 100 solving problems 1 through 9. Each line of input will consist of three numbers and a letter in the format contestant problem time L where L can be 'C', 'I', 'R', 'U' or 'E'. These stand for Correct, Incorrect, clarification Request, Unjudged and Erroneous submission. The last three cases do not affect scoring. Lines of input are in the order in which submissions were received.

Output:

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. Output will consist of a scoreboard sorted as previously described. Each line of output will contain a contestant number, the number of problems solved by the contestant and the time penalty accumulated by the contestant. Since not all of contestants 1-100 are actually participating, display only the contestants that have made a submission.

Source Code:

```
#include<stdio.h>

struct cricketer

{
```

```
int runs,wickets;
char name[25];
}player[100];
int main()
{
int i,n;
printf("Enter the no of cricket players\n");
scanf("%d",&n);
printf("Enter player info as name , runs scored , wickets taken\n");
for(i=0;i<n;i++)
{
scanf("%s %d %d",player[i].name,&player[i].runs,&player[i].wickets);
}
printf("\nNAME\t\tRUNS\t\tWICKETS\n");
for(i=0;i<n;i++)
{
printf("%s\t\t%d\t\t%d\n",player[i].name,player[i].runs,player[i].wickets);
}
return 0;
}
```

Output : 1

```
1 2 10 I
3 1 11 C
2 1 19 R
1 2 21 C
1 1 25 C
1 2 66
 1 3 11
```

Conclusion : In this Practical I learn to Implementation of contest Scoreboard.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 8	**Date: 08-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a Program on WERTYU.

Aim: Implementation of WERTYU.

Theory:

Problem:WERTYU

Create a program that will give a result of transferring one row to the right of the correct position of keys. For example, when the user entered “Q” or “q” the result will give the user “W” or “w” and so on so forth. The program must be able to decode a message in this sequence.

Inputs may contain numbers,spaces,upper and lower case,punctuations (except a back quote (‘)), but, shift,ctrl, and alt keys are not included.

Sample Input : O S, GOMR YPFSU/

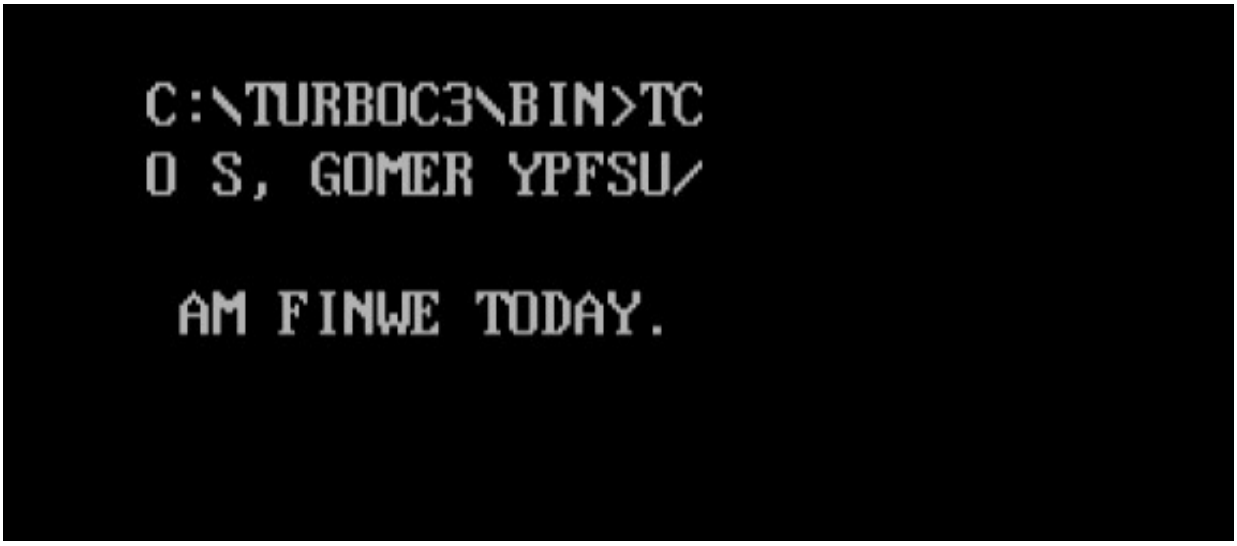
Sample Output : I AM FINE TODAY

Source Code :

```
#include <stdio.h>

#include <conio.h>

#include <string.h>

char set[60] = "`1234567890-=\\QWERTYUIOP[]ASDFGHJKL;\'ZXCVBNM,./";

int main()

{

char input[100];

gets(input);

int i, l = strlen(input);

strupr(input);

for (i = 0; i < l; i++)

{     int j;
```

```
for (j = 0; j < 47; j++)
{
if (set[j] == input[i] && input[i] != '`' && input[i] != 'Q' && input[i] != 'A' && input[i] != 'Z')
{
        printf("%c", set[j – 1]);
}
else if (input[i] == ' ')
{    printf(" ");
     break;
} } }
getch();
}
```

Output :



Conclusion : In this Practical I learn to Implementation of WERTYU.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 9	**Date: 19-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a Program on Permutation.

Aim: Implementation of the permutation.

Theory:

Permutation refers number of ways in which set members can be arranged or ordered in some fashion. The formula of permutation of arranging k elements out of n elements is

$$_{n}P\ k = n! / (n - k)!$$

Algorithm: This algorithm only focuses on permutation without going into details of factorial

START :

Step 1 → Define values for n and r

Step 2 → Calculate factorial of n and (n-r)

Step 3 → Divide factorial(n) by factorial(n-r)

Step 4 → Display result as permutation

STOP

Syntax:

procedure permutation()

Define n and r

P = factorial(n) / factorial(n-r)

DISPLAY Pend procedure.

Source Code:

```
#include <stdio.h>

int factorial(int n) {
```

```
int f;

for(f = 1; n > 1; n—)

{ f *= n;

   return f;     }

int npr(int n,int r)

{ return factorial(n)/factorial(n-r); }

int main()

{

int n, r;

n = 4;

r = 3;

printf("%dp%d = %d \n", n, r, npr(n,r));

return 0;

}
```

Output :

Conclusion : In this Practical I learn Implementation of the permutation.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 10	**Date: 19-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a Program on Automated Judge Script.

Aim: Study and implementation of Automated Judge Script.

Theory:

The judges from the programming contests are known to be very mean and very lazy. We, judges, want less work and more Wrong Answers! So, we'd like you to help us and write an automated judge script to judge solution runs from teams all over the world. All you have to do is write a program which receives the standard solution and a team output and gives as answer one of the following messages: "Accepted", "Presentation Error" or "Wrong Answer". We define each one as:

Accepted:

As we are very mean judges, we only want you to give "Accepted" as answer if the team output matches the standard solution integrally. That is, ALL characters must match and must be in the same order.

Presentation Error:

We want you to give "Presentation Error" if all NUMERIC characters match (and in the same order) but there is at least one non-numeric character wrong (or in wrong order). For instance, "15 0" and "150" would receive a "Presentation Error", whereas "15 0" and "1 0" would not (it would receive "Wrong Answer", see bellow).

Wrong Answer:

If the team output could not be classified as any of the two above, then you have no option but to give "Wrong Answer" as an answer!

Input File:

The input will consist of an arbitrary number of input sets. Each input set begins with a positive integer n < 100, alone in a line, which describes the number of lines of the standard solution. The next n lines contain the standard solution. Then there is a positive integer m < 100, alone in a line, which describes the number of lines of the team output. The next m lines contain the team output. The input is terminated by the end of file character. No line will have more than 120 character.

Output File: For each set you should output one of the following lines: Run #x: Accepted Run #x: Presentation Error Run #x: Wrong AnswerWhere x stands for the number of the input set (starting from 1).

Source Code:

```
#include<stdio.h>
//#include<conio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 13000
int main()
{
int run=1,m,n,i,j,k;
char ans[MAX],out[MAX],num[20],line[130],flag;
// clrscr();
while(fgets(num ,20 , stdin)) // replace all time 'fgats(line, 200, stdin)' with 'gets(num)'
for turbo c++
{
n=atoi(num);
if(!n) break;
memset(ans,0,sizeof(ans));
memset(out,0,sizeof(out));
for(i=0;i<n;i++)
{ fgets(line ,200 , stdin); strcat(ans,line); }
fgets(num, 30, stdin); m=atoi(num);
for(i=0;i<m;i++)
{ fgets(line, 200, stdin); strcat(out,line); }
if(!strcmp(ans,out)&&m==n) flag=1;
else
{
for(i=j=0;ans[i]&&out[j];i++)
if(ans[i]>='0'&&ans[i]<='9')
{
while(out[j]&&(out[j]<'0'||out[j]>'9')) j++;
```

```
if(ans[i]!=out[j]) { flag=-1; goto zap; }
j++;
}
flag=0;}
zap:
printf("Run #%d: ",run++);
if(flag>0) printf("Accepted\n");
else if(flag==0) printf("Presentation Error\n");
else printf("Wrong Answer\n");
//
}
getch();
return 0;
}
```

Output:

```
3
123
123
123
1
123123123
Run #1: Presentation Error
2
123
123
1
12312
Run #2: Wrong Answer
```

Conclusion : In this Practical I learn implementation of Automated Judge Script.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 11	**Date: 30-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program a file fragmentation

Aim: Implementation of file fragmentation

Theory:

File fragmentation:

File fragmentation is a term that describes a group of files that are scattered throughout a hard drive platter instead of one continuous location. Fragmentation is caused when information is deleted from a hard drive and small gaps are left behind to be filled by new data. As new data is saved to the computer it is placed in these gaps, if the gaps are too small the remainder of what needs to be saved is stored in remaining gaps.

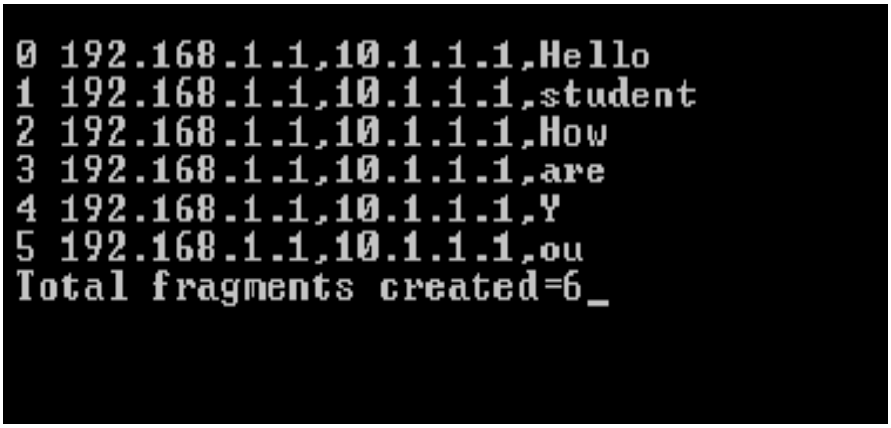
Fragmentation causes slow access time because read/write head accessing the data must find all fragments of a file before it can be opened or executed. If the hard drive has to do this for dozens or hundreds of different files as a program is opened, it can greatly decrease the overall performance of the computer. In the picture below is an example of file fragmentation, as can be seen the second example has other files and gaps in-between the continuous blue section.

Source Code:

```
#include <stdio.h>

#include <string.h>

int main()

{

Char ch;

int i=0,cnt=0;

FILE *fp;

Clrscr();

fp=fopen("in.txt","r");ch=fgetc(fp);

/*Assuming Source IP=192.168.1.1 and DestinationIP=10.1.1.1*/

Printf("\n 0 192.168.1.1,10.1.1.1,Hello",cnt++);

Printf("\n 1 192.168.1.1,10.1.1.1,Student",cnt++);
```

```
Printf("\n 2 192.168.1.1,10.1.1.1,How",cnt++);

Printf("\n 3 192.168.1.1,10.1.1.1,are", cnt++);

Printf("\n 4 192.168.1.1,10.1.1.1,Y",cnt++);

Printf("\n 5 192.168.1.1,10.1.1.1,ou",cnt++);

While(ch != EOF)

{

Printf("%c", ch);

i++;

if(i%5==0)

{

Printf("\n");

}

Ch=fgetc(fp);

}

Printf("\n Total fragments created=%d",cnt);

getch();

return 0;

}
```

Output:



Conclusion : In this Practical I learn Implementation of file fragmentation.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 12	**Date: 30-10-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on Vito's family.

Aim: Implement program on Vito's family.

Theory:

Vito's Family:

The world-known gangster Vito Dead stone is moving to New York. He has a very big family there, all of them living in Lamafia Avenue. Since he will visit all his relatives very often, he is trying to find a house close to them.

Problem:

Vito wants to minimize the total distance to all of them and has blackmailed you to write a program that solves his problem.

Input:

The input consists of several test cases. The first line contains the number of test cases. Note that several relatives could live in the same street number.

Output:

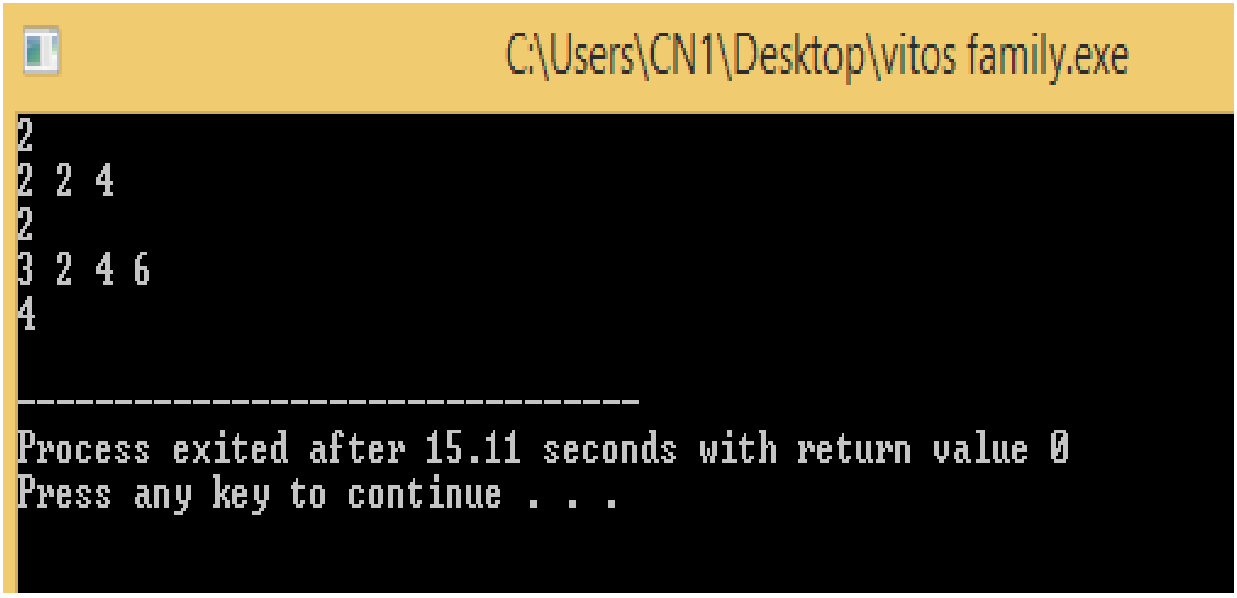
For each test case your program must write the minimal sum of distances from the optimal Vito's house to each one of his relatives. The distance between two street numbers s i and s j is d ij = |s i -s j |

Source Code:

```
#include<stdio.h>

#include<algorithm>

using namespace std;

int s[505];

int main(){

int t,n,i,k,add;

scanf("%d",&t);
```

```
while(t--){
scanf("%d",&n);
for(i=0;i<n;i++)
scanf("%d",&s[i]);
sort(s,s+n);
k=s[n/2];add=0;
for(i=0;i<n;i++)
add+=abs(s[i]-k);
printf("%d\n",add);
}
}
```

Output:



Conclusion : In this Practical I learn to Implement program on Vito's family.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 13	**Date: 08-11-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on Stack of Flapjack.

Aim: Implement program on Stack of Flapjack.

Theory:

Stack of Flapjack:

Stacks and Queues are often considered the bread and butter of data structures and find use in architecture, parsing, operating systems, and discrete event simulation. Stacks are also important in the theory of formal languages. This problem involves both butter and sustenance in the form of pancakes rather than bread in addition to a finicky server who flips pancakes according to a unique, but complete set of rules.

Given a stack of pancakes, you are to write a program that indicates how the stack can be sorted so that the largest pancake is on the bottom and the smallest pancake is on the top. The size of a pancake is given by the pancake's diameter. All pancakes in a stack have different diameters

Sorting a stack is done by a sequence of pancake ``flips''. A flip consists of inserting a spatula between two pancakes in a stack and flipping (reversing) the pancakes on the spatula (reversing the sub-stack). A flip is specified by giving the position of the pancake on the bottom of the sub-stack to be flipped (relative to the whole stack). The pancake on the bottom of the whole stack has position 1 and the pancake on the top of a stack of n pancakes has position n.

A stack is specified by giving the diameter of each pancake in the stack in the order in which the pancakes appear. For example, consider the three stacks of pancakes below (in which pancake 8 is the top-most pancake of the left stack):

8	7	2
4	6	5
6	4	8
7	8	4
5	5	6
2	2	7

The stack on the left can be transformed to the stack in the middle via flip(3). The middle stack can be transformed into the right stack via the command flip(1).

Source Code:

```
#include <stdio.h>
```

```
#include <stdlib.h>
int pancakes[31];
int pancakes2[31];
int n;
void preenche(char *a)
{
int i;
n=0;
i=0;
while(a[i]!='\n')
{
if((a[i]>=48)&&(a[i]<=57))
{
pancakes[n]=a[i]-48;
i++;
while((a[i]>=48)&&(a[i]<=57))
{
pancakes[n]=(pancakes[n]*10)+(a[i]-48);
i++;
};
n++;
}
else
{
i++;
};
};
};
void swap1(int a, int b)
{
int aux;
aux=pancakes[a];pancakes[a]=pancakes[b];
pancakes[b]=aux;
```

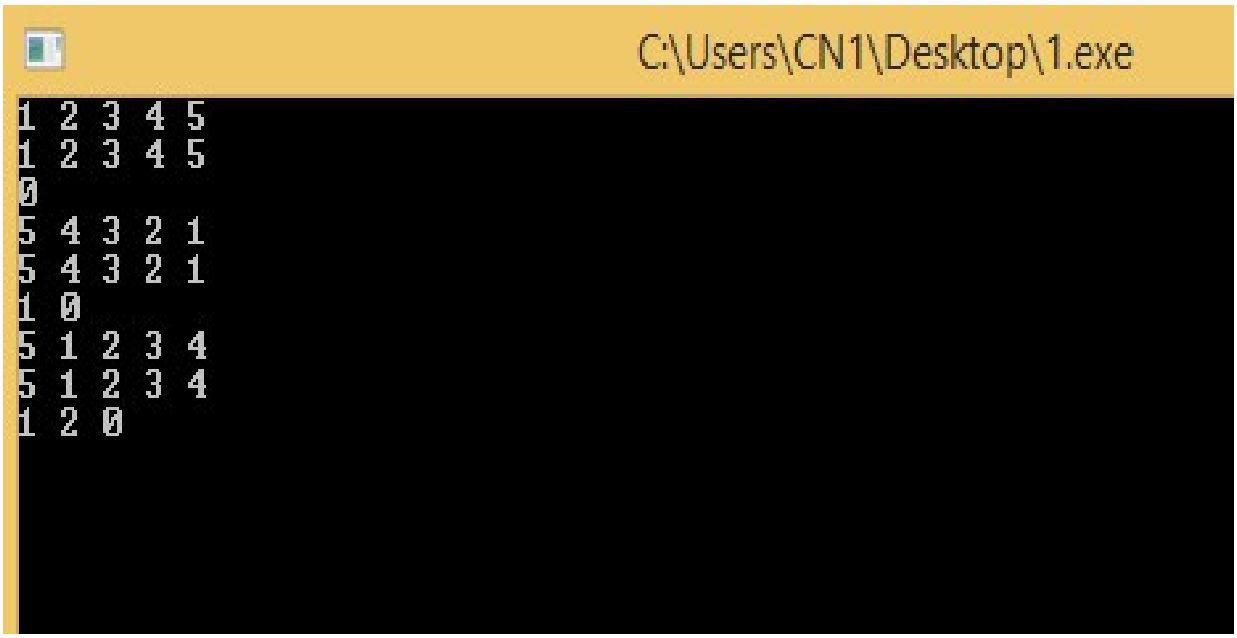
```
};
void inverte(int a)
{
int i, j;
for (i=0,j=a; i<j; i++,j--)
{
swap1(i,j);
};
};
void calcula()
{
int i;
int j;
int t;
for(i=0; i<(n-1); i++)
{
if(pancakes[0]==pancakes2[i])
{
inverte(n-1-i);
printf("%d ",i+1);
}
else
{
j=1;
t=-1;
while((j<(n-1-i)))
{
if(pancakes[j]==pancakes2[i])
{
t=j;
break;
};
j++;
```

```
};
if(t!=-1)
{
inverte(t);
inverte(n-1-i);
printf("%d ",n-t);
printf("%d ",i+1);
};
};
};
};void swap(int a, int b)
{
int aux;
aux=pancakes2[a];
pancakes2[a]=pancakes2[b];
pancakes2[b]=aux;
};
int heapify(int g, int a, int b)
{
int d1, d2;
if(a>b)
{
return 0;
};
d1=heapify(g,a*2,b);
d2=heapify(g,a*2+1,b);
if(d1>d2)
{
if(pancakes2[g+a-1]<d1)
{
swap(g+a-1,g+a*2-1);
};
}
```

```
else
{
if(pancakes2[g+a-1]<d2)
{
swap(g+a-1,g+a*2);
};
};
return pancakes2[g+a-1];
};
void heapsort()
{
int i;
for(i=0; i<n; i++)
{
heapify(i,1,n-i);
};
};
int main()
{
char *a;
int i;a=(char *)malloc(sizeof(char)*100);
a=fgets(a,100,stdin);
while(a)
{
preenche(a);
for(i=0; i<n; i++)
{
if(i>0)
{
printf(" ");
};
printf("%d",pancakes[i]);
pancakes2[i] = pancakes[i];
```

```
};
printf("\n");
heapsort();
calcula();
printf("0\n");
a=fgets(a,100,stdin);
};
return 0;
}
```

Output:



Conclusion : In this Practical I learn to Implement program on Stack of Flapjack.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 14	**Date: 08-11-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a Program on Bridge.

Aim :To implement the Bridge Program.

Theory:

The Bridge design pattern allows you to separate the abstraction from the implementation. It is a structural design pattern.

There are 2 parts in Bridge design pattern :

1. Abstraction

2. Implementation

- This is a design mechanism that encapsulates an implementation class inside of an interface class. The bridge pattern allows the Abstraction and the Implementation to be developed independently and the client code can access only the Abstraction part without being concerned about the Implementation part.
- The abstraction is an interface or abstract class and the implementor is also an interface or abstract class.
- The abstraction contains a reference to the implementor. Children of the abstraction are referred to as refined abstractions, and children of the implementor are concrete implementors. Since we can change the reference to the implementor in the abstraction, we are able to change the abstraction's implementor at run-time. Changes to the implementor do not affect client code.

"**In this C++ program, we'll see how to print reverse triangle bridge pattern in C++ using nesting of loops**."

Reverse Triangle Bridge Pattern looks like as:

```
A B C D E D C B A
A B C D   D C B A
A B C       C B A
A B           B A
A               A
```

To print this pattern, we can use ASCII codes of the corresponding characters to print them. Our program accepts the input of largest alphabet value in the pattern (e.g., C=3,E=5). Above

Pattern shows the constant width/spacing in both the reverse triangles. By using if, else if and else statement within the nesting of for loop gives the desired result.

Source Code:

```
#include<iostream.h>

#include<conio.h>

int main(){

int i,j,n;

cout<<"Enter Largest Alphabet Value(e.g c=3):";

cin>>n;

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

{     for(j=65;j<64+(2*n);j++)

      {

            if(j>=(64+n)+i)

                  cout<<(char)((64+n)-(j%(64+n)));

            else if(j<=(64+n)-i)

                  cout<<(char)j;

            else

                  cout<<" "; }

cout<<endl;

}

getch();

return 0; }
```

Output:

```
C:\TURBOC3\BIN>TC
Enter Largest Alphabet Value(e.g c=3):7
ABCDEFGFEDCBA
ABCDEF FEDCBA
ABCDE   EDCBA
ABCD     DCBA
ABC       CBA
AB         BA
A           A
```

Conclusion : In this Practical I learn to implement the Bridge Program.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 15	**Date: 23-11-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on shell sort.

Aim: Implement the shell sort.

Theory:

The idea of shell Sort is to allow exchange of far items. In shellSort, we make the array h-sorted for a large value of h. We keep reducing the value of h until it becomes 1. An array is said to be h-sorted if all sub lists of every h'th element is sorted.

Shell sort is the generalization of insertion sort which overcomes the drawbacks of insertion sort by comparing elements separated by a gap of several positions. In general, Shell sort performs the following steps.

1. **Step 1:** Arrange the elements in the tabular form and sort the columns by using insertion sort.
2. **Step 2:** Repeat Step 1; each time with smaller number of longer columns in such a way that at the end, there is only one column of data to be sorted.

Algorithm

Shell_Sort(Arr, n)

Step 1: SET FLAG = 1, GAP_SIZE = N

Step 2: Repeat Steps 3 to 6 while FLAG = 1 OR GAP_SIZE > 1

Step 3:SET FLAG = 0

Step 4:SET GAP_SIZE = (GAP_SIZE + 1) / 2

Step 5:Repeat Step 6 for I = 0 to I < (N -GAP_SIZE)

Step 6: if Arr[I + GAP_SIZE] > Arr[I]

SWAP Arr[I + GAP_SIZE], Arr[I]

SET FLAG = 0

Step 7: END

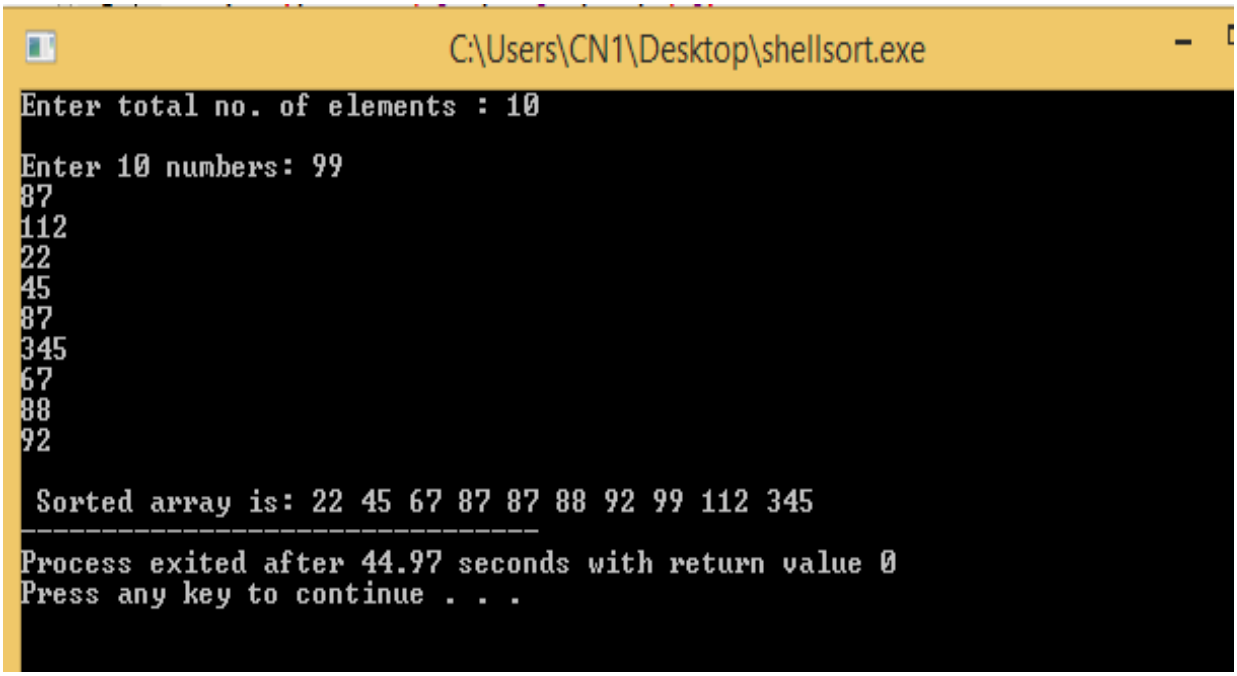
Source Code:

```
#include <stdio.h>
```

```
void shellsort(int arr[], int num)
{
int i, j, k, tmp;
for (i = num / 2; i > 0; i = i / 2)
{     for (j = i; j < num; j++)
      {
            for(k = j - i; k >= 0; k = k - i)
            {     if (arr[k+i] >= arr[k])
                        break;
                  else
                  {
                        tmp = arr[k];
                        arr[k] = arr[k+i];arr[k+i] = tmp;
                  }
 } } } }
int main()
{
int arr[30];
int k, num;
printf("Enter total no. of elements : ");
scanf("%d", &num);
printf("\nEnter %d numbers: ", num);
for (k = 0 ; k < num; k++)
{
scanf("%d", &arr[k]);
}
shellsort(arr, num);
printf("\n Sorted array is: ");
for (k = 0; k < num; k++)
```

```
printf("%d ", arr[k]);

return 0;

}
```

Output:



Conclusion : In this Practical I learn Implement the shell sort.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 16	**Date: 23-11-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a Program on Primary Arithmetic.

Aim: Study and implementation of Primary Arithmetic.

Theory:

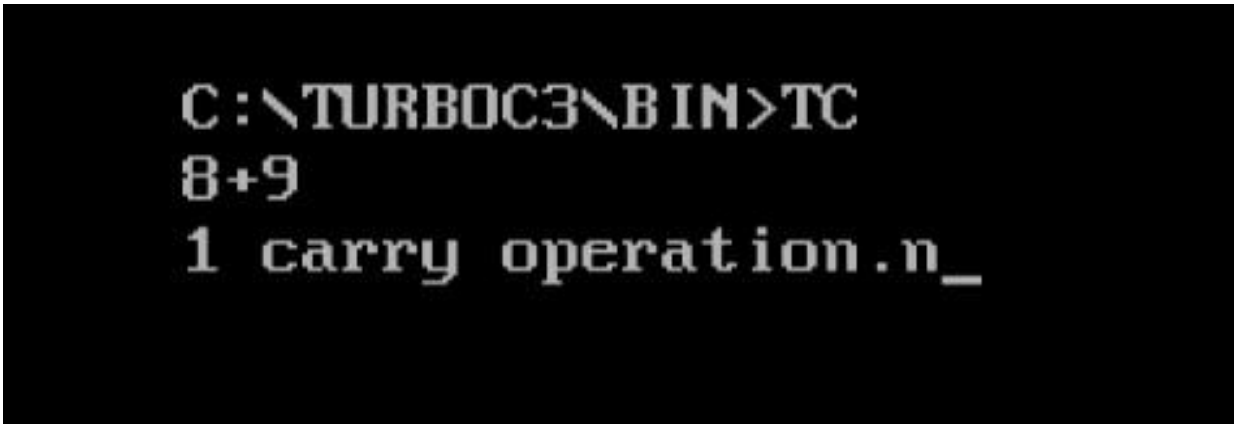
Children are taught to add multi-digit numbers from right to left, one digit at a time. Many find the "carry" operation, where a 1 is carried from one digit position to the next, to be a significant challenge. Your job is to count the number of carry operations for each of a set of addition problems so that educators may assess their difficulty.

Source Code:

```
#include <stdio.h>

#include <string.h>

long NumCarryAdd(long n1, long n2 ) {

long a,b,c,t;

c = 0;

t = 0;

while(1){

a=n1%10;

b=n2%10;

n1=n1/10;

n2=n2/10;

if((a+b+c)>=10){

t++;

c=1;

}

else c = 0;

if(n1==0 && n2==0)break;
```

```
}
return t;
}
int main(){long x, y, carry;
while(1){
scanf("%ld %ld", &x,&y);
if(x == 0 && y == 0) break;
carry = NumCarryAdd(x, y);
if(carry == 0) printf("No carry operation.n");
else if(carry==1)printf("1 carry operation.n");
else printf("%ld carry operations.n", carry);
}
return 0;
}
```

Output:



Conclusion: In this Practical I learn implementation of Primary Arithmetic.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 17	**Date: 28-11-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on Reverse and Add.

Aim: To implement the Reverse and Add.

Theory:

C program to reverse and add a number. This program reverses and add a number entered by a user and then print it on the screen. For example, if a user will enter 123 as input then 321 will be printed and sum the numbers and display the output.

Source code:

```
#include<stdio.h>

#include<conio.h>

int main()

{

int num, rev=0, n, digit;

clrscr();

printf("\n\t Enter the number: ");

scanf("%d", &num);

n = num;

while(n != 0)

{

digit = n % 10;

rev = rev*10 + digit;n = n/10;

}

printf("\n\t Reverse of entered number: %d", rev);

printf("\n\t sum = %d", rev + num);

getch();

return 0;
```

}

Output:

```
Enter the number: 123

Reverse of entered number:  321
sum = 444_
```

Conclusion : In this Practical I learn **i**mplement the Reverse and Add.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 18	**Date: 28-11-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write program on A Multiplication Game.

Aim: Study and Implementation of A Multiplication Program.

Theory:

Stan and Ollie play the game of multiplication by multiplying an integer p by one of the numbers 2 to 9. Stan always starts with p = 1, does his multiplication, then Ollie multiplies the number, then Stan and so on. Before a game starts, they draw an integer 1 < n < 4294967295 and the winner is who first reaches p $/l n.

Input and Output

Each line of input contains one integer number n. For each line of input output one line either Stan wins.

or

Ollie wins.

assuming that both of them play perfectly.

Sample input

162

17

34012226

Sample Output

Stan wins.

Ollie wins.

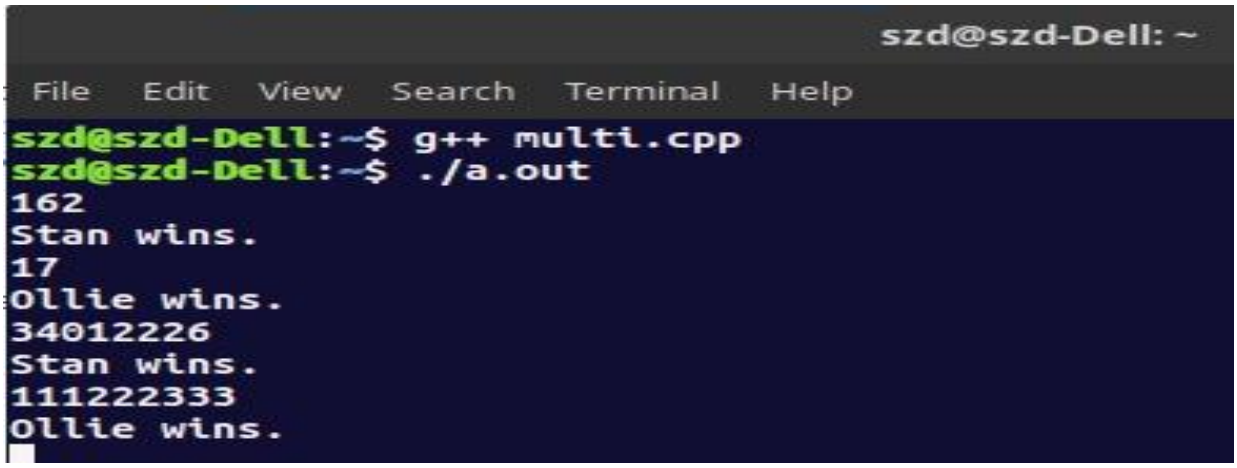
Stan wins.

Source Code:

```
#include<iostream>

using namespace std;

#define SET(a) memset(a,-1,sizeof(a))
```

```
#define CLR(a) memset(a,0,sizeof(a))

#define PI acos(-1.0)

#define MOD 1000000007

#define MX 100010

long long n;

int func(long long cur)

{ if(cur>=n) return 0;

int ret=0;

ret= ret | !func(cur*2);

ret= ret | !func(cur*9);

return ret;     }

int main()

{ //ios_base::sync_with_stdio(0);cin.tie(0);

int tc, kk=1;

string s;

while(cin>>n)

{ if(func(1))

cout<<"Stan wins.\n";

else cout<<"Ollie wins.\n";

} return 0; }
```

Output:



Conclusion : In this Practical I learn Implementation of A Multiplication Program.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 19	**Date: 02-12-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on Polynomial Coefficient.

Aim: Implement the Polynomial Coefficient.

Theory:

A Polynomial is an expression or a mathematical equation which contains variables and constants (also known as Coefficients). The different terms in the expression includes the operations of Addition, Non – Negative Integer Exponent, Subtraction and Multiplication. A polynomial is nothing but an algebraic expression. It is also famously known as arithmetic expression.

Source code:-

```
#include<stdio.h>

#include<math.h>

#include<conio.h>

int evaluate_polynomial(int arr[], int limit, int x)

{ int sum = 0, count;

for(count = limit; count >= 0; count--)

{ sum = sum + arr[count]*pow(x, count); }

return sum;

}

int main()

{ int array[30], degree, x_val, count, result;clrscr();

printf("\nEnter the Degree of Polynomial:\t");

scanf("%d", &degree);

printf("\nEnter the Co - Efficients:\n");

for(count = degree; count >= 0; count--)

{ printf("\nCo - Efficient of A[%d]: \t", degree);

  scanf("%d", &array[count]);
```

```
}
printf("\nThe Polynomial:\n\n");
for(count = degree; count >= 0; count--)
{ if(array[count] != 0)
{ printf("%dx^%d + ", array[count], count);
} }
printf("%d", array[count]);
printf("\n\nEnter the Value of X:\t");
scanf("%d", &x_val);
result = evaluate_polynomial(array, degree, x_val);
printf("\nEvaluation of Polynomial:\t%d\n", result);
getch();
return 0;
}
```

Output:

```
tushar@tusharsoni:~/Desktop$ gcc test.c -lm
tushar@tusharsoni:~/Desktop$ ./a.out

Enter the Degree of Polynomial: 4

Enter the Co - Efficients:

Co - Efficient of A[4]:         3

Co - Efficient of A[4]:         4

Co - Efficient of A[4]:         2

Co - Efficient of A[4]:         1

Co - Efficient of A[4]:         2

The Polynomial:

3x^4 + 4x^3 + 2x^2 + 1x^1 + 2x^0 + 0

Enter the Value of X:   3

Evaluation of Polynomial:       374
tushar@tusharsoni:~/Desktop$
```

Conclusion : In this Practical I learn to Implement the Polynomial Coefficient.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 20	**Date: 02-12-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program for how many fibs.

Aim: Implement program to find how many fibs.

Theory:

A series of whole numbers in which each number is the sum of the two preceding numbers. Beginning with 0 and 1, the sequence of Fibonacci numbers would be 0,1,1, 2, 3, 5, 8, 13, 21, 34, etc. using the formula n = n(-1) + n(-2), where the n(-1) means "the last number before n in the series" and n(-2) refers to "the second last one before n in the series."

Explanation

Fibs number = 0,1,1,2,3,5,8,13,21,34.

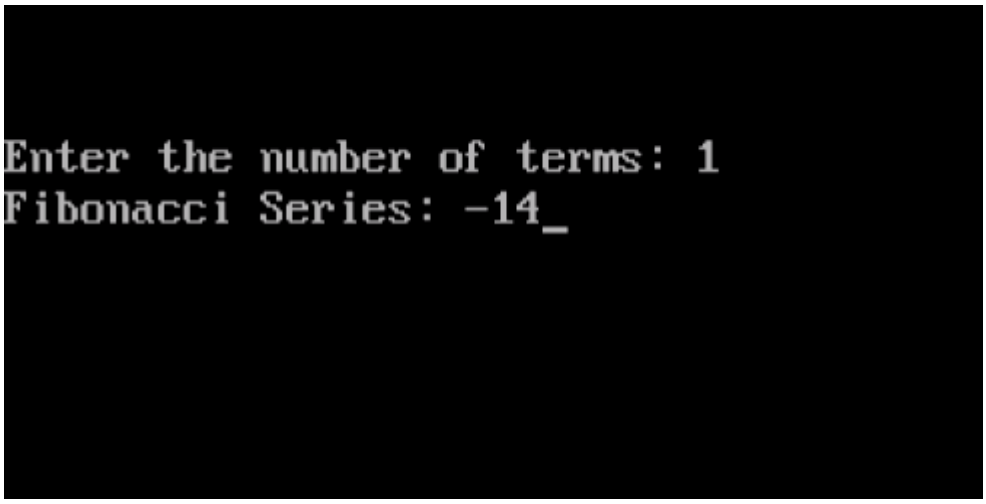
formula n = n(-1) + n(-2), where the n(-1) means "the last number before n in the series" and n(-2) refers to "the second last one before n in the series."

Sourse Code:

```
#include <stdio.h>

#include <conio.h>

int main()

{

int i, n, t1 = 0, t2 = 1, nextTerm;

printf("Enter the number of terms: ");

scanf("%d", &n);

printf("Fibonacci Series: ");

for (i = 1; i <= n; ++i)

{

printf("%d”,& t1);

nextTerm = t1 + t2;

t1 = t2;
```

```
t2 = nextTerm;}

getch();

return 0;

}
```

Output:



Conclusion : In this Practical I learn to Implement program to find how many fibs.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 21	**Date: 04-12-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program for counting.

Aim: Implement program for counting .

Theory:

C program to find the frequency of characters in a string: This program counts the frequency of characters in a string, i.e., which character is present how many times in the string. For example, in the string "code" each of the characters 'c,' 'd,' 'e,' and 'o' has occurred one time. Only lower case alphabets are considered, other characters (uppercase and special characters) are ignored. You can easily modify this program to handle uppercase and special symbols.

Source Code:

```
#include<stdio.h>

/* #include<conio.h> */

#include<stdlib.h>

#include<string.h>

int main()

{

char string[100];

int c = 0, count[26] = {0}, x;

//clrscr();

printf("Enter a string\n");gets(string);

while (string[c] != '\0') {

/** Considering characters from 'a' to 'z' only and ignoring others. */

if (string[c] >= 'a' && string[c] <= 'z') {

x = string[c] - 'a';

count[x]++;

}

c++;
```

```
}

for (c = 0; c < 26; c++)

printf("%c occurs %d times in the string.\n", c + 'a', count[c]);

//getch();

return 0;

}
```

Output:

```
Enter a string
shahzadalamchaudhary
a occurs 6 times in the string.
b occurs 0 times in the string.
c occurs 1 times in the string.
d occurs 2 times in the string.
e occurs 0 times in the string.
f occurs 0 times in the string.
g occurs 0 times in the string.
h occurs 4 times in the string.
i occurs 0 times in the string.
j occurs 0 times in the string.
k occurs 0 times in the string.
l occurs 1 times in the string.
m occurs 1 times in the string.
n occurs 0 times in the string.
o occurs 0 times in the string.
p occurs 0 times in the string.
q occurs 0 times in the string.
r occurs 1 times in the string.
s occurs 1 times in the string.
t occurs 0 times in the string.
u occurs 1 times in the string.
v occurs 0 times in the string.
w occurs 0 times in the string.
x occurs 0 times in the string.
y occurs 1 times in the string.
z occurs 1 times in the string.
lubuntu@lubuntu:~$
```

Conclusion : In this Practical I learn to implement program for counting Frequency of character.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 22	**Date: 04-12-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program for expression

Aim: Implement a program for expression

Theory:

A regular expression is a special string that describes a search pattern. Many of you have surely seen and used them already when typing expressions like ls(or dir) *.txt , to get a list of all the files with the extension txt. Regular expressions are very useful not only for pattern matching, but also for manipulating text. In SRMs regular expressions can be extremely handy. Many problems that require some coding can be written using regular expressions on a few lines, making your life much easier.

Many Topcoders believe that regular expressions are one of Java's main strengths over C++ in the arena. C++ programmers don't despair, regular expressions can be used in C++ too. There are several regular expression parsing libraries available for C+ +, unfortunately they are not very compatible with each other. Fortunately as a Topcoder in the arena one does not have to cope with all this variety of "not so compatible with one another" libraries. If you plan to use regular expressions in the arena you have to choose between two flavors of regex APIs: POSIX_regex and GNU_regex. To use these APIs the header file "regex.h" must be included. Both of these work in two steps – first there is a function call that compiles the regular expression, and then there is a function call that uses that compiled regular expression to search or match a string.

Source Code:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdio.h>

int main()

{

char expr1[] = "1+2*5+3";

int res = evaluate(expr1);

(res == -1)? cout << expr1 << " is " << "Invalid\n":
```

```
cout << "Value of " << expr1 << " is " << res << endl;


char expr2[] = "1+2*3";

res = evaluate(expr2);

(res == -1)? cout << expr2 << " is " << "Invalid\n":

cout << "Value of " << expr2 << " is " << res << endl;


char expr3[] = "4-2+6*3";

res = evaluate(expr3);

(res == -1)? cout << expr3 << " is " << "Invalid\n":cout << "Value of " << expr3 << " is " <<
res << endl;


char expr4[] = "1++2";

res = evaluate(expr4);

(res == -1)? cout << expr4 << " is " << "Invalid\n":

cout << "Value of " << expr4 << " is " << res << endl;

return 0; }
```

Output:

```
Value of 1+2*5+3 is 18
Value of 1+2*3 is 9
Value of 4-2+6*3 is 24
1++2 is Invalid
```

Conclusion : In this Practical I learn to implement a program for expression.

Godavari College Of Engineering, Jalgaon.

Subject Name: Competitive Programming - 1	**Teacher Name:** Prof. S. S. Shete
Practical No. : 23	**Date: 04-12-2020**
Class: T. Y. Computer Engineering.	**Roll No: 34**

Title: Write a program on Self-describing Sequence.

Aim: Implement program on Self-describing Sequence.

Theory:

What is Self-descriptive number?

A number is called as Self-descriptive when the position of digit represents the number of time it appears in that number.

The Self Theory emphasizes on the set of perceptions an individual has for himself and the perceptions of the relationships he has with others and the other aspects of life. Carl Rogers has contributed significantly towards the self theory.

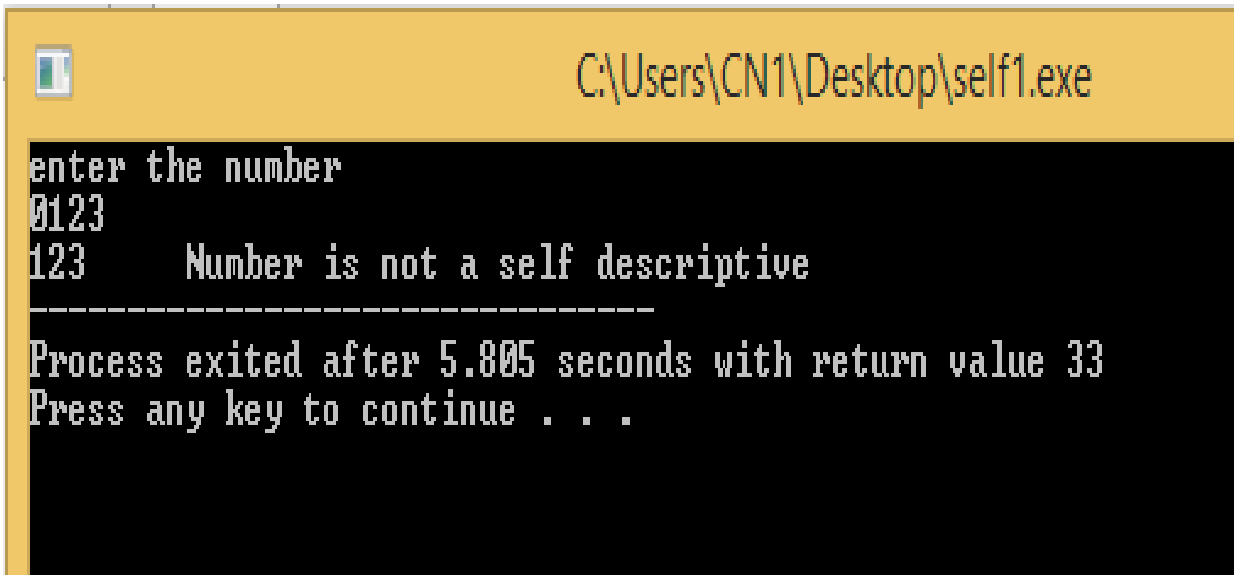
Source Code:

```
#include<stdio.h>

int check(int,int);

int getNumDigits(int);

int main()

{

int num=0,temp=0, digit=0,count=0;

printf("enter the number\n");

scanf("%d",&num);

temp=num;

int flag = 1;int numDigit = getNumDigits(temp);

while(temp>0)

{

digit=temp%10;

int count=check(num,numDigit);

if(count!= digit)
```

```
{
printf("\tNumber is not a self descriptive");
flag = 0;
break;
}
temp=temp/10;
numDigit--;
}
if(flag)
{
printf("\tNumber is a self descriptive");
}
}
/* check number of times the digit appear in number */
int check(int num,int digit)
{
int count=0;
while(num>0)
{
if(num%10 == digit)
{
count++;
}
num=num/10;
}
return count;
}
/* to check number of digits in number */
int getNumDigits(int num)
{
```

```
printf("%d",num);
int digits = 0;
while(num>0)
{
digits++;
num=num/10;}
return digits-1;
}
```

Output:



Conclusion : In this Practical I learn to Implement program on Self-describing Sequence.