TalentBattle

AMCAT AUTOMATA
Coding Questions

www.talentbattle.in

AMCAT Automata Questions

Easy:-

Q.1. Program to find the GCD of two numbers.

Example: Find the GCD.

Given numbers, 24 and 60.

Find the common factors- 24= 2,2,2,3 and 60 = 2,2,3,5

GCD = 2*2*3 = 12.

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int a,b,gcd;
  printf("\nEnter two numbers : ");
  scanf("%d %d",&a,&b);
  int i;
  for(i = 1; i <= a && i <= b; i++)
  {
   if((a % i == 0) && (b % i == 0))
   {
   gcd = i;
   }
 }
 printf("\nGCD of %d and %d is %d ",a,b,gcd);
 printf("\n");
return 0;
}
```

Q.2. Program to find GCD of two numbers using recursion.

Program:

```
#include <stdio.h>

int gcd(int a, int b)
{
  if (b != 0)
  return gcd(b, a % b);
  else
  return a;
}

int main()
{
  int a, b;
  printf(“Enter two numbers: “);
  scanf(“%d %d”, &a, &b);

  printf(“\nGCD of %d and %d is %d\n”, a, b, gcd(a,b));
  return 0;
```

Q.3. Program to find GCD of two numbers using Command Line Arguments.

Program:

```
#include<stdio.h>

int main(int x, char *y[])

{

  inta,b,small,i;

  a=atoi(y[1]);

  b=atoi(y[2]);

  small=a>b?b:a;

  for(i=small;i>=1;i–)
```

```
  {
   if((a%i==0)&&(b%i==0))
  {
   printf("%d",i);
   break;
} }
return 0;
}
```

Q.4. Program to find LCM of two numbers.

Program:

```
#include <stdio.h>

int main()
{
  int a, b, lcm;
  printf("\nEnter two numbers: ");
  scanf("%d %d", &a, &b);

  lcm = (a > b) ? a : b;

  while(1)
  {
   if( lcm % a == 0 && lcm % b == 0 )
   {
    printf("\nLCM of %d and %d is %d\n", a, b,lcm);
    break;
   }
  ++lcm;
 }
```

```
return 0;
}
```

Q.5. Program to find LCM of two numbers by finding their GCD.

Program:

```
// LCM of two numbers in C
#include <stdio.h>
int GCD(int a, int b)
{
  if (a == 0 || b == 0)
  return 0;
if (a == b)
return a;

if (a > b)
return GCD(a-b, b);
return GCD(a, b-a);
}

int LCM(int a, int b)
{
return (a*b)/GCD(a, b);
}

int main()
{
  int a,b;
  printf("\nEnter two numbers : ");
  scanf("%d %d",&a,&b);
  printf("\nLCM of %d and %d is %d \n", a, b, LCM(a, b));
  return 0;
}
```

Q.6. Check whether a given number is a prime or not.

Program:

```
// C program to check if the given  number is prime or not using for loop
```

```
#include<stdio.h>

int main()
{
  int n,i;
  printf("\nEnter the number : ");
  scanf("%d",&n);

  for(i = 2; i <= n/2; i++)
  {
   if(n % i ==0)
   {
    break;
   }
 }
 if(i > n/2)
 printf("\n%d is a Prime Number\n",n);
 else
 printf("\n%d is not a Prime Number\n", n);
 return 0;
}
```

Output:

Enter the number = 19

19 is a prime number.

Q.7. Program to check whether the given number is Armstrong or not.

(A number is an Armstrong number when the sum of nth power of each digit is equal to the number itself.)

Example:

1634 (here n=4) = $1^4+6^4+3^4+4^4$ = 1+1296+81+256 = 1634 => Armstrong number.

123(here n=3) = $1^3+2^3+3^3$ = 1+8+27 = 36 => Not Armstrong number.

Program:

```
// C program to check whether the given number is Armstrong or not

#include
#include

int main()
{
  int number, temp, remainder, result = 0, n = 0 ;

  printf(“Enter an integer: “);
  scanf(“%d”, &number);

  temp = number;

// Finding the number of digits

  while (temp != 0)
  {
   temp /= 10;
  ++n;
  }

  temp = number;

// Checking if the number is armstrong

 while (temp != 0)
 {
  remainder = temp%10;
  result += pow(remainder, n);
  temp /= 10;
 }

 if(result == number)
 printf(“%d is an Armstrong number\n”, number);
```

```
  else
  printf("%d is not an Armstrong number\n", number);

  return 0;
}
```

Output: Enter a number = 1634

1634 is an Armstrong number.

Enter a number = 156

156 is not an Armstrong number

Q.8. Program to check if a given number is a strong number or not.

(A strong number is a number in which the sum of the factorial of the digits is equal to the number itself.)

Example:

Strong number: 123 = 1!+2!+3! = 1+2+6 = 9 => Not a Strong number

145 = 1!+4!+5! = 1+24+120= 145 => a Strong number

Program:

```
// C program to check if a given number is a strong number or not

#include<stdio.h>
int main()
{
  int n,i;
  int fact,rem;
  printf("\nEnter a number : ");
  scanf("%d",&n);
  printf("\n");
```

```
 int sum = 0;
 int temp = n;
 while(n)
 {
  i = 1,fact = 1;
  rem = n % 10;

  while(i <= rem)
  {
   fact = fact * i;
   i++;
  }
  sum = sum + fact;
  n = n / 10;
 }
 if(sum == temp)
 printf("%d is a strong number\n",temp);
 else
 printf("%d is not a strong number\n",temp);

 return 0;
}
```

Q.9. Program to check whether a number is Automorphic number or not

(An automorphic number is a number whose square ends with the number itself.)

Example: $5^2 = 25$ => Automorphic number

$7^2 = 49$ => Not Automorphic number

Program:

```
// C program to check whether the number is automorphic or not

#include<stdio.h>
bool isAutomorphic(int N)
```

```
{
  int sq = N * N;

  while (N > 0)
  {
   if (N % 10 != sq % 10)
   return false;

// Reduce N and square
   N /= 10;
   sq /= 10;
 }

  return true;
}

int main()
{
//Fill the code
  int N;
  scanf("%d",&N);
  isAutomorphic(N) ? printf("Automorphic") : printf("Not Automorphic");
  return 0;
}
```

Output:

5 => Automorphic number

Q.10. Program to check whether a number is Abundant number or not.

(An abundant number is a number for which the sum of its proper divisors is greater than the number itself.)

Example:

Given a number, 12. The divisors of 12 are 1,2,3,4 and 6.

The sum of divisors of 12 is 16.

12<16. 12 is an Abundant number.

Program:

// C program to check whether a number is an abundant number or not

```
#include<stdio.h>
int main()
{
//fill the code
  int num;
  int temp;
  scanf("%d",&num);
  int sum = 0;
  for(int i = 1; i < num; i++)
  {
   if(num % i == 0)
   {
    sum = sum + i;
   }
  }
  if(num < sum)
  printf("Abundant Number");
  else
 printf("Not Abundant Number");
 return 0;
}
```

Q.11. Program to find prime numbers in a given range.

Program:

// C program to find prime numbers in a given range

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

```
 int a, b, i, flag;
 printf("\nEnter start value : ");
 scanf("%d",&a);
 printf("\nEnter end value : ");
 scanf("%d",&b);
 printf("\nPrime Numbers between %d and %d : ", a, b);
 while (a < b)
{
 flag = 0;
 for(i = 2; i <= a/2; ++i)
 {
  if(a % i == 0)
  {
   flag = 1;
   break;
  }
 }
if (flag == 0)
printf("%d ", a);
++a;
}
printf("\n");
return 0;
}
```

Output: Enter start value = 10

Enter end value = 50

Prime number between 10 and 50 = 11 13 17 19 23 29 31 37 41 43 47

Q.12. Program to generate Fibonacci series upto n value.

(Fibonacci series is a series in which each number is the sum of the last two preceding numbers. The first two terms of a Fibonacci series are 0 and 1.)

Example:

Generate Fibonacci series up to 200.

The first two terms are 0 and 1.

0 + 1 = 1

1 + 1 = 2

1 + 2 = 3

2 + 3 = 5

3 + 5 = 8

5 + 8 = 13

8 + 13 = 21

13 + 21 = 34

21 + 34 = 55

34 + 55 = 89

55 + 89 = 144

144 + 89 = 233

The Fibonacci series up to 200 is given as follows.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.

Program:

```
// C program to generate fibonacci series upto n value
#include<stdio.h>
int main()
{
  int sum = 0, n;
  int a = 0;
```

```
  int b = 1;
  printf("Enter the nth value: ");
  scanf("%d", &n);
  printf("Fibonacci series: ");
  while(sum <= n)
  {
   printf("%d ", sum);
   a = b; // swap elements
   b = sum;
   sum = a + b; // next term is the sum of the last two terms
 }
 return 0;
}
```

Q.13. Program to print solid square star pattern.

Program:

```
// C program to print solid square star pattern

#include
int main()
{
  int i, j,n;
  scanf(“%d”,&n);
  for(i = 0; i <n; i++)
  {
   for(j = 0; j < n; j++)
   {
    printf(“*”);
```

```
  }
  printf(“\n”);
 }
return 0;
}
```

Output:

```
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
```

Q.14. Program to print hollow square star pattern.

Program:

```
// C program to print hollow square star pattern
#include <stdio.h>
int main()
{
  int i, j, n;
  scanf(“%d”,&n);
  for (i = 1; i <= n; i++)
  {
   for (j = 1; j <= n; j++)
   {
    if (i==1 || i==n || j==1 || j==n)
    printf(“*”);
    else
    printf(” “);
   }
    printf(“\n”);
  }
 return 0;
}
```

Output:

```
* * * * * * *
*           *
*           *
*           *
*           *
*           *
* * * * * * *
```

Medium:-

Q.15. Program to print all the Armstrong numbers between the two intervals.

Program:

```
// C program to print the Armstrong numbers between the two intervals

#include <stdio.h>
#include <math.h>
int main()
{
int start, end, i, temp1, temp2, remainder, n = 0, result = 0;
printf("Enter start value and end value : ");
scanf("%d %d", &start, &end);
printf("\nArmstrong numbers between %d an %d are: ", start, end);
for(i = start + 1; i < end; ++i)
{
  temp2 = i;
  temp1 = i;

  while (temp1 != 0)
  {
   temp1 /= 10;
   ++n;
   }
```

```
while (temp2 != 0)
{
 remainder = temp2 % 10;
 result += pow(remainder, n);
 temp2 /= 10;
}
if (result == i) {
printf("%d ", i);
}n = 0;
result = 0;
}
printf("\n");
return 0;
}
```

Output:

Enter start value and end value = 100 500

Armstrong numbers between 100 and 500 = 370 371 407

Q. 16. Program to convert the given binary number into decimal number.

Example:

Consider the binary number, 1111

$1 * 2^0 = 1$

$1 * 2^1 = 2$

$1 * 2^2 = 4$

$1 * 2^3 = 6$

Decimal number = 1 + 2 + 4 + 6 = 15

1111 in binary form is represented as 15 in decimal.

Program:

```
// C program to convert a binary number into decimal number

#include
#include

int binary_to_decimal(long int n)
{
  int decimal = 0, i = 0, remainder;
  while (n!=0)
 {
   remainder = n%10;
   n /= 10;
   decimal += remainder*pow(2,i);
   ++i;
 }
return decimal;
}

 int main()
{
  long int n;
  printf("Enter a binary number: ");
  scanf("%ld", &n);
  printf("\nDecimal number : %d\n ", binary_to_decimal(n));
  return 0;
}
```

Output:

Enter a binary number = 11111111

Decimal number = 255

Q.17. Program to convert a number from decimal to binary.

Example: The binary equivalent of decimal number 15.

15 / 2 = 7, rem = 1

7 / 2 = 3, rem = 1

3 / 2 = 1, rem = 1

1 / 2 = 0, rem = 1

Binary equivalent of 15 is 1111.

Program:

```
#include <stdio.h>

long int decimal_to_binary(int n)

{

  long int binary = 0;

  int remainder, i, flag = 1;

  for(i = 1; n != 0; i = i * 10)

  {

  remainder = n % 2;

  n /= 2;

  binary += remainder * i;

  }

  return binary;

}

int main()

{

 int n;
```

```
 printf("Enter a decimal number: ");
 scanf("%d", &n);
 printf("Equivalent binary number: %d\n", decimal_to_binary(n));
 return 0;
}
```

Output:

Enter a decimal number = 255

Equivalent binary number = 11111111

Q.18. Program to convert a number from decimal to octal.

Example: Decimal number 200 has to be converted to octal.

200 / 8 = 25, rem = 0

25 / 8 = 3, rem = 1

3 / 8 = 0, rem = 3

The equivalent octal number of the decimal number 200 is 310.

Program:

```
// C program to convert a number from decimal to octal

#include <stdio.h>
#include <math.h>

int decimal_to_octal(int decimal);
int main()
{
  int decimal;

  printf("\nEnter a decimal number: ");
  scanf("%d", &decimal);
```

```
   printf("\nEquivalent octal number : %d\n", decimal_to_octal(decimal));

  return 0;
}

int decimal_to_octal(int decimal)
{
  int octal = 0, i = 1;

  while (decimal != 0)
  {
   octal += (decimal % 8) * i;
   decimal /= 8;
   i *= 10;
  }

return octal;
}
```

Output:

Enter a decimal number = 200

Equivalent octal number = 310

Q.19. Program to convert a number from octal to decimal.

Example: Octal number 1907 has to be converted to decimal.

$7 * 8^0 = 7$

$0 * 8^1 = 0$

$9 * 8^2 = 576$

$1 * 8^3 = 512$

512 + 576 + 0 + 7 = 1095. The decimal equivalent of octal number 1907 is 1095.

Program:

```
// C program to convert a number from octal to decimal

#include
#include

long int octal_to_decimal(int octal)
{
  int decimal = 0, i = 0;

  while(octal != 0)
  {
   decimal += (octal%10) * pow(8,i); // multiplying with powers of 8
   ++i;
  octal/=10;  // Divide by 10 to make it as decimal
 }
i = 1;
return decimal;
}
int main()
{
  int octal;

  printf("\nEnter an octal number: ");
  scanf("%d", &octal);

  printf("\nDecimal Equivalent : %d\n",octal_to_decimal(octal));

  return 0;
}
```

Output:

Enter an octal number = 350

Decimal equivalent = 232

Q.20. Program to reverse a number.

Example: Input: 13579

Output: 97531

Program:

```
// C program to reverse a number

#include <stdio.h>
int main()
{
  int n, rev = 0, rem;
  printf("\nEnter a number : ");
  scanf("%d", &n);
  printf("\nReversed Number : ");
  while(n != 0)
  {
   rem = n%10;
   rev = rev*10 + rem;
   n /= 10;
  }

  printf("%d\n", rev);

  return 0;
}
```

Output: Enter a number = 12345

Reversed number = 54321

Q.21. Program to print palindrome pyramid pattern using numbers.

Program:

```
/* C program for Palindrome pyramid pattern printing using numbers */
#include<stdio.h>

#include<stdlib.h>
int main()
{
  int i,j,k,l,n;
  printf("Enter the number of rows : ");
  scanf("%d",&n);
  for(i = 1; i <= n; i++)
  {
   for(k = 1; k <= i; k++)
   {
    printf("%d ",k);
   }
   for(l = i-1; l >= 1; l–)
  {
   printf("%d ",l);
   }
   printf("\n");
 }
 return 0;
}
```

Output:

```
1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

Q.22. Program to print palindrome pyramid pattern using alphabets.

Program:

```
/* C program for Palindrome pyramid pattern printing using alphabets */
#include<stdio.h>
#include<stdlib.h>
int main()
```

```
{
  int i,j,k,l,n;
  printf("Enter the number of rows : ");
  scanf("%d",&n);
  for(i = 1; i <= n; i++)
  {
   for(k = 1; k <= i; k++)
   {
    printf("%c ",(k + 65 – 1));
   }
    for(l = i-1; l >= 1; l–)
   {
   printf("%c ",(l + 65 – 1));
   }
   printf("\n");
  }
 return 0;
}
```

Output:

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

Average:-

Q.23. Remove vowels from a string and return the string with consonants.

Example: Remove vowels from string.

Vowels: a, e, i, o, u

Input string: Talent Battle

Output string: Tlnt Bttl

Program:

```
#include <stdio.h>
int check_vowel(char);
int main()
{
  char s[100], t[100];
  int c, d = 0;
  gets(s);
  for(c = 0; s[c] != '\0'; c++)
  {
   if(check_vowel(s[c]) == 0)
   {
    t[d] = s[c];
    d++;
   }
  }
  t[d] = '\0';
  strcpy(s, t);
  printf("%s\n", s);
  return 0;
}
int check_vowel(char ch)
{
  if (ch == 'a' || ch == 'A' || ch == 'e' || ch == 'E' || ch == 'i' || ch == 'I' || ch
=='o' || ch=='O' || ch == 'u' || ch == 'U')
  return 1;
  else
  return 0;
}
```

Q.24. Program to find all the triplets with the given sum.

Example: Consider the array: arr [] = {0, -1, 2, -3, 1}. The given sum is -2. In the given array, the triplets with sum = -2 are {0, -3, 1} and {-1, 2, -3}.

Program:

// C program to find all the triplets with the given sum

```
#include
void find_all_triplets(int arr[], int n, int sum)
{
  for (int i = 0; i < n – 2; i++)
 {
  for (int j = i + 1; j < n – 1; j++)
  {
   for (int k = j + 1; k < n; k++)
   {
   if (arr[i] + arr[j] + arr[k] == sum)
   {
    printf(“%d,%d,%d\n”,arr[i],arr[j],arr[k]);
}}}}}

int main()
{
  int n, sum;
  printf(“\nEnter the number of elements : “);
  scanf(“%d”,&n);
  int arr[n];
  printf(“\nInput the array elements : “);
  for(int i = 0; i < n; i++)
  {
   scanf(“%d”,&arr[i]);
  }
  printf(“\nEnter the sum value : “);
  scanf(“%d”,&sum);
  printf(“\nThe triplets are \n “);
  find_all_triplets(arr, n, sum);
return 0;
}
```

Output:

Enter the number of element: 5

Input the array elements : 0 -1 2 -3 1

Enter the sum value: -2

The triplets are : -3 -1 2 and -3 0 1

Q.25. Program to find all the triplets with the given sum using sorting technique.

Program:

```
// C++ program to find all the triplets with the given sum

#include <bits/stdc++.h>
using namespace std;

void find_all_triplets(int arr[], int n, int sum)
{
  sort(arr, arr + n);
for (int i = 0; i < n – 1; i++)
{
  int l = i + 1;
  int r = n – 1;
  int x = arr[i];
  while (l < r) {
  if (x + arr[l] + arr[r] == sum)
  {
   printf(“%d %d %d\n”, x, arr[l], arr[r]);
   l++;
   r–;
  }

  else if (x + arr[l] + arr[r] < sum)
  l++;
else
  r–;
  }
 }
}
```

```
int main()
{
  int n, sum;
  cout << "\nEnter the number of elements : ";
  cin >> n;
  int arr[n];
  cout << "\nInput the array elements : ";
  for(int i = 0; i < n; i++)
  {
   cin >> arr[i];
  }
  cout << "\nEnter the sum value : ";
  cin >> sum;
  cout << "\nThe triplets are \n ";
  find_all_triplets(arr, n, sum);
  return 0;
}
```

Output: Enter the number of element: 5

Input the array elements : 0 -1 2 -3 1

Enter the sum value: -2

The triplets are : -3 -1 2 and -3 0 1

Q.26. Program to find all the triplets with the given sum using hashing technique.

Program:

```
#include <bits/stdc++.h>
using namespace std;

void find_all_triplets(int arr[], int n, int sum)
{
  for (int i = 0; i < n – 1; i++) {
```

```
  unordered_set<int> s;
  for (int j = i + 1; j < n; j++) {
  int x = sum – (arr[i] + arr[j]);
  if (s.find(x) != s.end())
  printf("%d %d %d\n", x, arr[i], arr[j]);
  else
  s.insert(arr[j]);
  }
 }
}

int main()
{
  int n, sum;
  cout << "\nEnter the number of elements : ";
  cin >> n;
  int arr[n];
  cout << "\nInput the array elements : ";
  for(int i = 0; i < n; i++)
 {
  cin >> arr[i];
 }
 cout << "\nEnter the sum value : ";
 cin >> sum;
 cout << "\nThe triplets are \n ";
 find_all_triplets(arr, n, sum);
 return 0;
}
```

Output: Enter the number of element: 5

Input the array elements : 0 -1 2 -3 1

Enter the sum value: -2

The triplets are : -3 -1 2 and -3 0 1

Q.27. Program to find the frequency of characters in a string.

Program:

```
// C program to find the frequency of characters in the string

#include <stdio.h>
#include <string.h>
#define MAX 50

int main()
{
char str[MAX];
int i, len;
int frequency[26];

printf("Enter the string: ");
gets(str);

len = strlen(str);

for(i=0; i<26; i++)
{
frequency[i] = 0;
}

for(i=0; i<len; i++)
{
if(str[i]>='a' && str[i]<='z')
{
frequency[str[i] – 97]++;
}
else if(str[i]>='A' && str[i]<='Z')
{
frequency[str[i] – 65]++;
}
}

printf("\nFrequency of characters : \n");
for(i=0; i<26; i++)
{
```

```
if(frequency[i] != 0)
{
printf("%c –> %d\n", (i + 97), frequency[i]);
}
}

return 0;
}
```

Output:

Enter the string: hello world

Frequency of characters:

d→ 1

e→ 1

h→ 1

l→ 1

o→ 1

r→ 1

w→ 1

Q.28. Program to find all the roots of a quadratic equation.

Program:

```
// C program to find all the roots of a quadratic equation
```

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

int main()
{
double a, b, c, discriminant, root1, root2, realPart, imaginaryPart;

printf(“Enter coefficients a, b and c: “);
scanf(“%lf %lf %lf”,&a, &b, &c);

discriminant = b*b-4*a*c;

// condition for real and different roots
if (discriminant > 0)
{
// sqrt() function returns square root
root1 = (-b+sqrt(discriminant))/(2*a);
root2 = (-b-sqrt(discriminant))/(2*a);

printf(“root1 = %.2lf and root2 = %.2lf”,root1 , root2);
}

//condition for real and equal roots
else if (discriminant == 0)
{
root1 = root2 = -b/(2*a);

printf(“root1 = root2 = %.2lf;”, root1);
}

// if roots are not real
else
{
realPart = -b/(2*a);
imaginaryPart = sqrt(-discriminant)/(2*a);
printf(“root1 = %.2lf+%.2lfi and root2 = %.2f-%.2fi”, realPart, imaginaryPart,
realPart, imaginaryPart);
}
```

```
return 0;
}
```

Output:

Enter coefficients a, b and c : 1 2 3

Root1 : -1+1.41421 and Root2: -1+1.41421.

Q.29. Program to find the largest palindrome in an array.

Program:

```
// c program to find Largest palindrome in an array

#include

int check_palindrome(int n)
{
  int div = 1;
  while (n / div >= 10)
  div *= 10;

  while (n != 0)
  {
   int first = n / div;
   int last = n % 10;

// If first and last digits are not same then return false
   if (first != last)
   return -1;

// Removing the leading and trailing digits from the number
n = (n % div) / 10;

// Reducing divisor by a factor of 2 as 2 digits are dropped
   div = div / 100;
   }
```

```
   return 1;
 }
int large_palindrome(int A[], int n)
{

// Sort the array
 for(int i=0; i<=n; i++)
 {
  for(int j=i; j<= n; j++)
  {
   if(A[i] >A [j])
   {
   int temp = A[i];
   A[i] = A[j];
  A[j] = temp;
   }
 }
}

for(int i=0; i<n; i++)
{
 printf("%d ", A[i]);
}

 for (int i = n – 1; i >= 0; –i)
 {
  if (check_palindrome(A[i]) == 1)
  return A[i];
 }
 return -1;
}
int main()
{
  int a[15], n, i;
  printf("Enter the number of entries: \n");
  scanf("%d", &n);
  printf("Enter the elements: \n");
  for(i=0; i<n; i++)
  scanf("%d", &a[i]);
  printf("\n Largest Palindrome: %d", large_palindrome(a, n));
```

```
 return 0;
}
```

Output: Enter the number of entries: 5

Enter the elements: 12321

11111

4545

687

95

Largest palindrome : 12321

Q.30. Program for solid diamond pattern using stars.

Program:

```
/* C program – solid diamond pattern printing using stars */
#include <stdio.h>
int main()
{
int n, c, k, space = 1;
printf("Enter the number of rows\n");
scanf("%d", &n);
space = n – 1;
for (k = 1; k <= n; k++)
{
  for (c = 1; c <= space; c++)
  printf(" ");
space–;
for (c = 1; c <= 2*k-1; c++)
printf("*");
printf("\n");
```

```
}
space = 1;

for (k = 1; k <= n – 1; k++)
{
for (c = 1; c <= space; c++)
printf(” “);
space++;
for (c = 1 ; c <= 2*(n-k)-1; c++)
printf(“*”);

printf(“\n”);
}
return 0;
}
```

Output:

```
   *

  * *

 * * *

* * * *

* * * * *

* * * * *

 * * * *

  * * *

   * *

    *
```

Q.31. Program for Solid Half Diamond pattern printing using stars.

Program:

```
/* C program – solid half diamond pattern printing using stars */

#include

int main()
{
  int i, j, space, k = 0, n;
  printf(“\nEnter the number of rows : “);
  scanf(“%d”,&n);
  for(int i=1;i<=n;i++)
  {
   for(int j=1;j<=n-i;j++)
  {
   printf(” “);
  }
  for(int j=1;j<=i;j++)
  {
   printf(“*”);
  }

  printf(“\n”);
 }

for(int i=n-1;i>0;i–)
{
  for(int j=1;j<=n-i;j++)
  {
  printf(” “);
  }
 for(int j=1;j<=i;j++)
 {
  printf(“*”);
 }
 printf(“\n”);
```

```
 }
}
```

Output:

```
*

* *

* * *

* * * *

* * * * *

* * * *

* * *

* *

*
```

Q.32. Program to print palindrome pyramid pattern using numbers.

Program:

```
/* C program for Palindrome pyramid pattern printing using numbers */
#include<stdio.h>
#include<stdlib.h>
int main()
{
int i,j,k,l,n;
printf("Enter the number of rows : ");
scanf("%d",&n);
for(i = 1; i <= n; i++)
{
for(k = 1; k <= i; k++)
```

```
{
printf("%d ",k);
}
for(l = i-1; l >= 1; l–)
{
printf("%d ",l);
}
printf("\n");
}
return 0;
}
```

Output:

```
1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

Q.33. **Program to print palindrome pyramid pattern using alphabets.**

Program:

```
/* C program for Palindrome pyramid pattern printing using alphabets */
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int i,j,k,l,n;
  printf("Enter the number of rows : ");
  scanf("%d",&n);
  for(i = 1; i <= n; i++)
  {
   for(k = 1; k <= i; k++)
   {
    printf("%c ",(k + 65 – 1));
   }
    for(l = i-1; l >= 1; l–)
   {
   printf("%c ",(l + 65 – 1));
```

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

Output:

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

Q.34. Program to print palindrome pyramid pattern using numbers.

Program:

```
#include <stdio.h>
int main()
{
  int i, j, k, space, n;
  scanf("%d", &n);
  printf(" ");
  for (i=1; i<=n; i++)
  {

  for (j=1; j<=n-i; j++)
  printf(" ");

  for (j=1,k=2*i-1; j<=2*i-1; j++,k–)
  {
   if (j <= k)
   printf("%d", j);
  else
  printf("%d", k);
 }
 printf("\n");

 printf(" ");
```

```
 }
return 0;
}
```

Output:

```
        1

     1  2  1

   1 2 3 2 1

  1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1
```

Q.35. Program to print palindrome pyramid pattern using numbers & stars.

Program:

```
#include<stdio.h>
int main()
{
int n, i, j, space, count = 1, num = 0, star = 8;
scanf("%d", &n);
space = n;
for (i = 1; i <= n; i++)
{
for (j = 1; j <= star; j++)
if(i + j <= star + 1)
printf("*");
num++;
for (j = 1; j <= i; j++)
{
printf("%d", num);
if (i > 1 && count < i)
{
printf("*");
count++;
```

```
}
}
for (j = 1; j <= star; j++)
if(i + n <= j + n)
printf("*");
printf("\n");
space–;
count = 1;
}
return 0;
}
```

Output:

```
********1********

*******2*2*******

******3*3*3******

*****4*4*4*4*****

****5*5*5*5*5****

***6*6*6*6*6*6***
```

Recently asked AMCAT Automata Questions

(Some of the recently asked AMCAT Automata questions with their answers are discussed here. These questions are of easy and hard difficulty levels.)

Q.36. Find the number of all possible triplets in the array that can form the triangle (condition is a + b > c).

Program:

```
#include<stdio.h>

int arr[100], n, n1, n2, i, j, k;
```

```
int a,b,c;
int main()
{
  scanf(“%d”,&n);
  for(i=0;i<n;i++)
  scanf(“%d”,&arr[i]);
  for(i=0;i<n-2;i++)
  {
   a=arr[i];
   for(j=i+1;j<n-1;j++)
   {
    b=arr[j];
    for(k=j+1;k<n;k++)
   {
     c=arr[k];
    if( ((a + b)>c) && ((a + c)>b) && ((b + c)>a) )
   {
    printf(“%d  %d  %d  “,a,b,c);
    printf(“Yes\n”);
   }
   else
   {
```

```
   printf(“%d  %d  %d  “,a,b,c);
   printf(“No\n”);
   }
  }
 }
 }
return 0;
}
```

Q.37. Print all the prime numbers which are below the given number separated by comma.

Input: 50

Program:

```
#include"stdio.h"
int main()
{
int n,i,j,ct=0;
scanf(“%d”,&n);
for(i=2;i<=n;i++)
{
ct=0;
for(j=2;j<i;j++)
{
if(i%j==0)
```

```
{ ct=1; break; }
}
if(ct==0)
{
if(i>2)
printf(“, “);
printf(“%d”,i);
}
}
return 0;
}
```

Output: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

Q.38. Program to find the GCD of two Integers.

Program:

```
#include”stdio.h”
int main()
{
  int  m, n, i, gcd;
  scanf(“%d %d”, &m, &n);
  for(i=1; i <=m && i <= n; ++i)
  {
```

```
// Checks if i is factor of both integers
  if(m%i==0 && n%i==0)
  gcd = i;
 }
 printf(" %d", gcd);
 return 0;
}
```

Q.39. Program to find out the sum of digits of a given number.

Program:

```
#include "stdio.h"
void main()
{
  long n, t, digit;
  int sum = 0;
  scanf("%ld", &n);
  t = n;
  while (n > 0)
  {
  digit = n % 10;
  sum = sum + digit;
  n /= 10;
  }
```

```
  printf(“%d”, sum);
}
```

Q.40. Print the pattern if the input is 5.

Program:

```
#include<stdio.h>
int main()
{
int i, j, k, l=1, N, d, r, count=0;
 scanf(“%d”, &N);
 for(i=1; i<=N; i++)
 {
  k=1;
  d=i%2;
  r=l+i-1;
  for(j=0;j<i;j++)
  {
   if(d==0)
   {
    printf(“%d”,r);
    r–;
    if(k<i)
   {
     printf(“*”);
```

```
    k=k+1;
  }
   l++;
continue;
 }
 printf("%d",l);
  l++;
  if(k<i)
  {
   printf("*");
   k=k+1;
  }
 }
 printf("\n");
}
return 0;
}
```

Output:

1

3*2

4*5*6

10*9*8*7

11*12*13*14*15

Q.41. Mooshak the mouse has been placed in a maze. There is a huge chunk of cheese somewhere in the maze. The maze is represented as a two-dimensional array of integers, where 0 represents walls.1 represents paths where Mooshak can move and 9 represents the huge chunk of cheese. Mooshak starts in the top left corner at 0.
Write a method is Path of class Maze Path to determine if Mooshak can reach the huge chunk of cheese. The input to is Path consists of a two-dimensional array and for the maze matrix. the method should return 1 if there is a path from Mooshak to the cheese and 0 if not Mooshak is not allowed to leave the maze or climb on walls.
Example: 8 by 8(8*8) matrix maze where Mooshak can get the cheese.

Test Case 1:

Input: [[0,0,0],[9,1,1],[0,1,1]]

Expected return value: 0

Explanation: The piece of cheese is placed at(1,0) on the grid Mooshak can move from (0,0) to (1,0) to reach it or can move from (0,0) to (0,1) to (1,1) to (1,0)

Program:

```
#include”stdlib.h”

#include”stdio.h”

int path(int maze[3][3]);

int main()

{

 int i,j,maze[3][3], result=0;

 for(int i=0;i<3;i++)

  {

   for(int j=0;j<3;j++)

   scanf (“%d”, &maze[i][j]);
```

```
  }
  printf(“input\n”);
  for(int i=0;i<3;i++)
  {
   for(int j=0;j<3;j++)
  {
   printf(“%d “,maze[i][j]);
  }
  printf(“\n”);
 }
printf(“%d”, path(maze));
return 0;
}
int path(int m[3][3])
{ for(static int i=0;i<3;i++)
{ for(int j=0;j<3;j++)
{ if((m[i][j]==1))
int path(m[3][3]);
if(m[i][j]==0)
return 0;
if(m[i][j]==9)
return 1;
```

```
}

}

}
```

Output:

```
1 0 1 1 1 0 0 1

1 0 0 0 1 1 1 1

1 0 0 0 0 0 0 0

1 0 1 0 9 0 1 1

1 1 1 0 1 0 0 1

1 0 1 0 1 1 0 1

1 0 0 0 0 1 0 1

1 1 1 1 1 1 1 1
```

Q.42. Program to print all distinct elements of given input arrays. Also print the total of the distinct elements.

Input:

Arr1 = {1,2,3,4,5}

Arr 2 = {2,6,8,10}

Program:

```
#include"stdio.h"

int Not_common (int *arr1, int *arr2, int l1, int l2)

{

 int count =0, flag1, i, j;
```

```
for(i=0; i<l1; i++)
{
 flag1=0;
 for(j=0; j<l2; j++)
 {
  if(arr1[i] == arr2[j])
 {
  flag1=1;  break;
 }}
if(flag1 ==0)
{
count++;
printf("%d,", arr1[i]);
}
}
for(i=0; i<l2; i++)
{
 flag1=0;
 for(j=0; j<l1; j++)
 {
  if(arr2[i] == arr1[j])
 {
```

```
 flag1=1;
 break;
 }}
 if(flag1 ==0)
 {
  count++;
  printf("%d,", arr2[i]);
 }}
return count;
}
 int main()
 {
  int len1=3,len2=3, result, i, j;
  int arr1[10],arr2[10];
  scanf("%d %d", &len1, &len2);
  for(i=0; i<len1; i++)
  scanf("%d", &arr1[i]);
  for(i=0; i<len2; i++)
  scanf("%d", &arr2[i]);
  result = Not_common (arr1,arr2,len1,len2);
  printf("\n %d", result);
  return 0;
```

```
}
```

Output: {1,3,4,5,6,8,10} total is 7

Q.43. Swap corresponding elements of the array.

(Traversing the array and swapping i and i + 1 element.)

Program:

```
#include
using namespace std;
void swap(int *p1, int *p2){
//Function swaps two element
int t;
t = *p1;
*p1 = *p2;
*p2 = t;
}
int * swapArr(int *arr, int len){
int *t;
for(int i = 0; i < len - 1; i++){
//Sends a pointer to two coresponding elements
swap(arr + i, arr + i + 1);
}
return arr;
```

```
}
int main(){
int n, *p;
//Input size of array
cin >> n;
int arr[n];
for(int i = 0; i < n; i++){
cin >> arr[i];
}
//p gets the pointer to array after swapping
p = swapArr(arr, n);
cout << endl;
for(int i = 0; i < n; i++){
cout << *(arr + i) << " ";
}
cout << endl;
return 0;
}
```

Q.44. Print the below pattern

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Program:

```
#include
using namespace std;
void printPascal(int n)
{
int x = n;
for (int line = 1; line <= n; line++)
{
for(int j = x; j > 0; j--)
cout << " ";
x--;
int C = 1;
for (int i = 1; i <= line; i++)
{
cout << C << " ";
C = C * (line - i) / i;
```

```
}
cout << endl;
}
}
int main()
{
int x;
//Input number of rows
cin >> x;
printPascal(x);
return 0;
}
```

Output:

```
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
```

Q.45. Print the below pattern where Input n=6.

1111112

3222222

3333334

5444444

5555556

7666666

Program:

```
#include

using namespace std;
void patternprint(int);

int main()

{

int num;

cin>>num;

patternprint(num);

return 0;

}
void patternprint(int num)

{

int x = 1;

for( int i = 0 ; i < num ; i++ )

{
```

```
if(i % 2 == 0)
{
for( int j = 1 ; j <= num ; j++)
cout << x;
cout << x + 1;
}
else
{
cout << x + 1;
for( int j = 1 ; j <= num ; j++)
cout << x;
}
cout << "\n";
x++;
}}
```

Output:

```
1111112
3222222
3333334
5444444
5555556
7666666
```

Q.46. Programming Pattern to Print 2*N Number of rows for input Pattern?

3

44

555

6666

555

44

3

Program:

```
#include <iostream>
using namespace std;
int main()
{
  int n=4,num=n-1;
  for(int i=1;i<=n;i++)
  {
   for(int j=1;j<=i;j++)
   cout<<num;
   num++;
   cout<<endl;
  }
  num--;
```

```
 for(int i=n;i>=1;i--)
 {
 for(int j=1;j<=i;j++)
 cout<<num;
 num--;
 cout<<endl;
 }
return 0;
}
```

Q.47. Amcat Trapezium pattern Solution.To print the trapezium pattern.

Example : we have num=4

Program:

```
#include<iostream>
using namespace std;
int main(){
 int n=4,num=1,i=1,space=0,k=1,number=n;
 for(i=0;i<n;i++)
 {
 for(int j=1;j<=space;j++)
 {
  cout<<"-";
```

```
}
for(int m=1;m<2*n-space;m++)
{
 if(m%2==0)
 cout<<"*";
 else
 cout<<num++;
}
cout<<"*";
for(int l=1;l<2*n-space;l++)
{
 if(l%2==0)
 cout<<"*";
 else
 {
  cout<<k+number*number;
  k++;
  }
}
number--;
space=space+2;
cout<<endl;
```

```
  }
 return 0;
}
```

Output:

1*2*3*4*17*18*19*20

- -5*6*7*14*15*16

- - - -8*9*12*13

- - - - - -10*11

Q.48. Find the number of occurrences of input num2 in input num1 and return it with function int isOccured(int num1, int num2).

Input: num1 and num2

such that 0 <= num1 <= 99999999 and 0 <= num2 <= 9

Example:

Input: num1= 199294, num2= 9

Output: 3

Test Case 1:

Input:

1222212

2

Expected Output:

5

Test Case 2:

Input:

1001010

0

Expected Output:

4

Its solution is like the frequency count of number.

Q.49. Print all prime no which are below then given input no separated by comma .

Input:

20

Output:

2,3,5,7,11,13,17,19

Program:

```
<iostream>

using namespace std;

int main() {

 int n=20,count,k=0,arr[10];

 for(int i=2;i<n;i++)

 {count=0;

 for(int j=2;j<i;j++)

 {
```

```
 if(i%j==0)
 {
  count++;
  break;
 }}
if(count==0)
{
 arr[k]=i;
 k++;
}}
 cout<<"Prime no are \n";
for(int i=0;i<k-1;i++)
{
 if(i!=k-2)
 cout<<arr[i]<<",";
 else
 cout<<arr[i];
 }
return 0;
}
```

Q.50. You have 3 dice , you need to take input from the user what's the sum of three dice he wants .Program to find all the possible outcome of input.

Example:

Input- 5
Output- 6
Explanation- (1,2,2) (2,2,1) (2,1,2) (1,1,3) (1,3,1) (3,1,1)

Input- 2
Output- 0
Explanation- Minimum sum value on the dice is 3 , so it not possible to come sum equal to 2

Input- 3
Output- 1
Explanation- (1,1,1)

Program:

```
#include<iostream.h>

int main()

{

  int x,count=0;

  cin>>x;

  for(int i=1;i<=6;i++)

  {

   for(int j=1;j<=6;j++)

   {

    for(int k=1;k<=6;k++)

   {

     if(x==(i+j+k))

     count++;

    }
```

```
  }
}
 cout<<count;
return 0;
}
```

Q. 51. Program to Display Armstrong Number between two Intervals

Program:

```
#include <stdio.h>
int main()
{
  int n1, n2, i, temp, num, rem;
  printf("Enter two numbers(intervals): ");
  scanf("%d %d", &n1, &n2);
  printf("Armstrong numbers between %d an %d are: ", n1, n2);
  for(i=n1+1; i<n2; ++i)
  {
   temp=i;
   num=0;
   while(temp!=0)
   {
    rem=(temp%10);
```

```
   num+=rem*rem*rem;
    temp/=10;
  }
  if(i==num)
 {
 printf("%d ",i);
 }
}
return 0;
}
```

Q.52. Program to Display Prime Numbers between Two Intervals.

Program:

```
<stdio.h>
int main()
{
int n1, n2, i, j, flag;
printf("Enter two numbers(intevals): ");
scanf("%d %d", &n1, &n2);
printf("Prime numbers between %d and %d are: ", n1, n2);
for(i=n1+1; i<n2; ++i)
{
```

```
flag=0;
for(j=2; j<=i/2; ++j)
{
if(i%j==0)
{
flag=1;
break;
}
}
if(flag==0)
printf("%d ",i);
}
return 0;
}
```

Q.53. program to find out prime factors of given number.

Algorithm

1. Find factor of a given number

2. Apply prime number program to check whether factor is prime or not

Program:

```
main()
{
```

```
int num,i=1,j,k;
printf("\nEnter a number:");
scanf("%d",&num);
while(i<=num)
{
k=0;
if(num%i==0) /* Finding factor */
{
j=1;
while(j<=i) /* Check prime or not */
{
if(i%j==0)
k++;
j++;
}
if(k==2)
printf("\n%d is a prime factor",i);
}
i++;
}
}
```

Q.54. Write a function that accepts a sentence as a parameter, and returns the same with each of its words reversed. The returned sentence should have 1 blank space between each pair of words. Demonstrate the usage of this function from a main program.

Example:

Input: “jack and jill went up a hill”

Output: “kcaj dna llij tnew pu a llih”

Program:

```
#include <iostream>

#include<string.h>

using namespace std;

void strRev(char *arr, int startIndex, int endIndex) {

for(int i=endIndex; i>=startIndex; i--)

{

cout<< arr[i];

}

cout<<" ";

}

int main() {

char st[70]="jack and jill went up hill";

int p;

p = strlen(st);

int startIndex = 0;

//int wordCount = 0;
```

```
for(int j=0;j<=p;j++)

{

if(j == p) {

strRev(st, startIndex, j -1);

// wordCount++;

}

if(st[j] == ' ') {

strRev(st, startIndex, j);

wordCount++;

startIndex = j + 1;

}

}

//cout<< "Word Count " << wordCount;

return 0;

}
```

Q.55. Given an array of integers, write a function that returns true if there is a triplet (a, b, c) that satisfies $a^2 + b^2 = c^2$.

Example:

Input: arr [] = {3, 1, 4, 6, 5}

Output: True

There is a Pythagorean triplet (3, 4, 5).

Input: arr [] = {10, 4, 6, 12, 5}

Output: False

There is no Pythagorean triplet.

Program:

```
#include <iostream>

using namespace std;

// Returns true if there is Pythagorean triplet in ar[0..n-1]

bool isTriplet(int ar[], int n)

{

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

 {

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

  {

   for (int k=j+1; k<n; k++)

  {

// Calculate square of array elements

   int x = ar[i]*ar[i], y = ar[j]*ar[j], z = ar[k]*ar[k];

   if (x == y + z || y == x + z || z == x + y)

   return true;

   }

  }

}

// If we reach here, no triplet found
```

```
  return false;
 }
int main()
{
  int ar[] = {3, 1, 4, 6, 5};
  int ar_size = sizeof(ar)/sizeof(ar[0]);
  isTriplet(ar, ar_size)? cout << "Yes": cout << "No";
  return 0;
}
```