Note: The thought process written in red ink is just for reference as to how to tackle a problem while in exam hall.

Q1

Convert the following:

(i) 153 from base 10 to hexadecimal

\[
\begin{array}{c|c|c}
16 & 153 \\
-16 & -9 \\
\hline
& 9 \\
\end{array}
\]

\[ \therefore (153)_{10} = (99)_{16} \]

(ii) 100110111011 from base 2 to decimal

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
2^9 & 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
\hline
\end{array}
\]

\[= (1 \times 2^9) + (1 \times 2^8) + (1 \times 2^7) + (1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)\]

\[= 524,285 \]

\[\therefore (100110111011)_2 = (524,285)_{10} \]

(iii) 143 from base 8 to decimal

\[
\begin{array}{c|c|c|c|c}
8^2 & 8^1 & 8^0 \\
1 & 4 & 3 \\
\hline
\end{array}
\]

\[= 1 \times 8^2 + 4 \times 8^1 + 3 \times 8^0 = 99 \]

\[\therefore (143)_8 = (99)_{10} \]
Q2.2. Write a program using recursion to calculate the value of \( z = x^y \).

Thought process:

\[
\begin{align*}
x^0 &= 1 \\
x &= x^{y \div 2} \times x^{y \div 2} & \quad \text{case } y \text{ even} \\
\text{if } y \text{ odd} & \quad x &= x \times x^{y - 1}
\end{align*}
\]

Using the above logic,

\[
x^y = x \times x^{y-1} 
\]

\[
\text{findpower(base, power) = base + findpower(base, power-1)}
\]

Recursion.

```c
#include <stdio.h>

int findpower(int base, int power)
{
    if(power>0)
    {
        return base*findpower(base,power-1);
    }
    return 1;
}

int main()
{
    int base, power, z;
    printf("Enter the number x: ");
    scanf("%d",&base);
    printf("Enter the power y: ");
    scanf("%d",&power);
    z = findpower(base,power);
    printf("The value of \( z \) is : \%d",z);
    return 0;
}
```

Q2.2. Write a function to reverse a three digit number.

Thought process:

Given a number 'num', we try to form its reverse 'rev' and print rev.

\[
\begin{align*}
\text{Iteration 1:} & \quad \text{num} = 456 \quad \text{rem} = \text{num} \div 10 \\
& \quad \text{rev} = 000 + \text{rem} = 006 \\
& \quad \text{num} = 45 \quad \text{rem} = \text{num} \div 10 \\
& \quad \text{rev} = 006 + \text{rem} = 006 \\
\text{Iteration 2:} & \quad \text{num} = 45 \quad \text{rem} = \text{num} \div 10 \\
& \quad \text{rev} = 006 + \text{rem} = 006 \\
& \quad \text{num} = 065 \quad \text{rem} = \text{num} \div 10 \\
& \quad \text{rev} = 006 + \text{rem} = 006 + 5 = 65
\end{align*}
\]
#include <stdio.h>

int main()
{
    int num, rem, rev=0;
    printf("Enter the number x: ");
    scanf("%d", &num);

    while(num>0)
    {
        rem = num % 10;
        rev = rev * 10 + rem;
        num = num / 10;
    }

    printf("The reversed number is: \xd", rev);
    return 0;
}

### Question 2.c) Print the following pattern:

```
1 0 0 0 0 0
2 0 1 0 0 0
3 0 0 1 0 0
4 0 1 1 0 0
5 0 0 0 0 0
```

### Thought process:

- We have to print 2 things:
  1. Space for loop.
  2. I/O depending on some pattern if/else.

To print:

(i) Space:
- Space is printed everywhere when \( j < i \)

(ii) I/O:
- \( i \) even \( \rightarrow \) 0
- \( i \) odd \( \rightarrow \) 1
- \( j \) even \( \rightarrow \) 0
- \( j \) odd \( \rightarrow \) 0

\[ \begin{array}{ccccccc}
1 & 2 & 3 & 4 & 5 \\
5 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 1 & 0 & 0 & 0 \\
3 & 0 & 0 & 1 & 0 & 0 \\
2 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]
```c
#include <stdio.h>

int main()
{
    int i, j;
    for (i = 5; i >= 1; i--)
    {
        for (j = 5; j >= 1; j--)
        {
            if (i % 2 == j % 2)
                printf(" "); // printing space
            else if (i % 2 == j % 2)
                printf("0");
            else
                printf("1");
        }
        printf("\n"); // changing line
    }
    return 0;
}
```
Q3

28 March 2020

2:19

Exercise 3.4: Write a program to find the frequency of digits in a set of numbers and remove duplicates from it. Example: \( A = \{1, 2, 3, 4, 2, 5, 2\} \) \( \Rightarrow \) frequency of 2 is 2 and resultant array is \( A = \{1, 2, 3, 4, 5\} \).

**Thinking process:**

- There are 3 parts to this problem:
  1. Finding frequency
  2. If an element is counted, don't count it again.

- Checker array: Each element in checker array will tell about the status of corresponding element.

- **Initial case:**
  
  \[
  \begin{array}{c|c|c|c|c}
  \hline
  \text{arr} & 1 & 2 & 1 & 2 \\
  \hline
  \text{checker} & 1 & 1 & 1 & 1 \\
  \hline
  \end{array}
  \]

- Finally after running program:
  
  \[
  \begin{array}{c|c|c|c|c}
  \hline
  \text{arr} & 1 & 2 & 1 & 2 \\
  \hline
  \text{checker} & 2 & 2 & -1 & -1 \\
  \hline
  \end{array}
  \]

- \( \text{checker}[j] = 1 \Rightarrow \) initially every element is present at least once.

- \( \text{checker}[0] = 2 \Rightarrow \text{arr}[0] \) is repeated twice
  - \( \text{checker}[1] = 2 \Rightarrow \text{arr}[1] \) is repeated twice
  - \( \text{checker}[2] = -1 \Rightarrow \text{arr}[2] \) is repeated element. So, discard it.
  - \( \text{checker}[3] = -1 \Rightarrow \text{arr}[3] \) is repeated element. So, discard it.

- So, to find resultant array, we take only those \( i \) whose \( \text{checker}[i] = -1 \)
  - i.e. only those elements which are not repeated.

- Resultant array:
  
  \[
  \{1, 2\}
  \]

- Working of program:
  
  i) Loop from \( i = 0 \) to \( i = n - 1 \)
  
  j) Loop from \( j = i + 1 \) to \( j = n - 1 \) - Comparing all elements ahead of \( i \) with \( i \) to find frequency.
check only elements whose checker is not -1. If checker is -1, it is already marked and counted.

Finding resultant array

```c
for (i = 0; i < n; i++)
    { 
        if (checker[i] != -1)
            { 
                if (arr[i] == arr[j])
                    { 
                        if (checker[j] != -1)
                            checker[j] = -1;
                        checker[i] += 1;
                    }
            }
```

Transferring only elements whose checker != -1, i.e., if (checker != -1)

```c
for (i = 0; i < n; i++)
    { 
        if (checker != -1)
            { 
                resultants[k] = arr[i];
                k++;
            }
```

```c
}
```

```c
//Printing frequency
for (i = 0; i < n; i++)
    { 
        if (checker[i] != -1)
            { 
                print("Frequency of %d is %d", arr[i], checker[i]);
                resultants[i] = arr[i]; // Creating a final array with no repetition
                k = 1;
            }
```

```c
//Printing final array
print("Final array:");
for (i = 0; i < n; i++)
    print("%d", resultants[i]);
return 0;
```

Que 3. Refer Dec 2019 solved paper on RAIT ACM website.

FE_CP May 19 Page 6
Q4. a) Write a program to check whether entered string is palindrome or not without using inbuilt string function.

Thought process:

- Given any string, we have to compare first half with corresponding 2nd half.
- Let's consider 2 cases:
  1. n=5 (odd) \( \Rightarrow \) \( n/2 = 5/2 = 2 \) \( \Rightarrow \) \( i = 0 \text{ to } i < n/2 \)
  2. n=4 (even) \( \Rightarrow \) \( i = 0, 1 \)

Case 1: n=5 (odd) \( \Rightarrow \) \( n/2 = 5/2 = 2 \) \( \Rightarrow \) \( i = 0 \text{ to } i < n/2 \)
- We don't have to compare central element in odd, as it will always be a single element.

- Checking string element.
- for any \( i \), the corresponding element from last is \( i = (n-i-1) \)

To check if palindrome or not, we define an integer variable ispal.

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

int main()
{
    char string[100];
    int n,i,ispal;

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

    for(i=0; string[i] != '\0'; i++)
    {
        if (string[i] == string[n-i-1])
            ispal=1;
        else
            ispal=0,
            break;
    }

    if (ispal == 1)
        printf("Palindrome\n");
    else
        printf("Not a palindrome\n");

    return 0;
}
```

```c
0 1 2 3 4
i=0
n-1 = n-1 = n-1 = n-1
i = 0, 1, 2, 3
```

```c
0 1 2 3
i=0, 1
```

© FE_CP_May19 Student Chapter
#include <stdio.h>
#include <string.h>

int main()
{
    char string[100];
    int n, i, ispal;
    printf("Enter string\n");
    getstr(string);
    for (i = 0; string[i] != '\0'; i++)
    {
        n++; //finding out length of string
    }
    for (i = 0; i < n / 2; i++)
    {
        if (string[i] == string[n - i - 1])
            ispal = 1;
        else
            ispal = 0;
            break;
    }
    if (ispal == 1)
        printf("Palindrome\n");
    else
        printf("Not Palindrome\n");
    return 0;
}

Question 4.6) Factorial using call by reference.

Thought Process: Call by reference: passing address of an argument in a formal parameter.

Here, thus we will have to find address of number whose factorial we have to find.

The value stored at that address i.e., number will be used to calculate factorial.
Q5. Write a program to find sum of the diagonal elements of a matrix.

Example:

\[
\begin{bmatrix}
1 & 2 & 3 \\
5 & 9 & 8 \\
& & \\
1 & 4 & 7
\end{bmatrix}
\Rightarrow \text{sum} = 3 + 9 + 1 = 13.
\]

Thought process:

- Matrix \(\rightarrow\) 2D array 'n' elements \(\Rightarrow\) matrix is a square matrix
- Diagonal elements will be found at positions where \(i=j\).

\[
\begin{array}{cccc}
& & & \text{Sum} = 0 \\
& & 1 & \\
& 2 & & \\
& & 3 & \vdots \\
0 & & & \\
1 & & & \\
2 & & & \\
& & & \\
& & & \\
\end{array}
\]

\[
i = 0 \text{ to } n-1 \\
j = 0 \text{ to } n-1
\]

C program:

```c
#include <stdio.h>

int main()
{
    int matrix[20][20], i, j, sum = 0, n;
    printf("Enter the size of matrix n: \\
"); scan("%d", &n);
    printf("Enter elements of matrix\n");
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
        {
            scanf("%d", &matrix[i][j]);
            if(i == j)
            {
                sum = sum + matrix[i][j]; //calculating sum as we get elements
            }
        }
    }
    printf("The sum is \%d \", sum);
    return 0;
}
```

Question 6.b: Write a program to calculate the sum of series:

\[ S = \frac{1}{n^1} - \frac{1}{n^2} + \frac{1}{n^3} - \frac{1}{n^4} + \frac{1}{n^5} + \ldots \]

```c
#include <stdio.h>

int main()
{
    int a, b, c, big;
    printf("Enter three numbers : ");
    scanf("%d %d %d", &a, &b, &c);
    big = (a > b) ? (a > c) ? a : c : (b > c) ? b : c;
    printf("The biggest number is : %d", big);
    return 0;
}
```
S = \frac{x}{1!} - \frac{x}{2!} + \frac{x}{3!} - \frac{x}{4!} + \frac{x}{5!} + \cdots + \frac{x}{n!}

Thought process:

→ There are \( n \) terms in the series.

\[
S = x - \frac{x}{2!} + \frac{x}{3!} - \frac{x}{4!} + \cdots + \text{n terms} = x \left( \frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!} + \cdots + \text{n terms} \right)
\]

\[
\therefore S = x \left( \sum_{i=1}^{n} \frac{(-1)^{i+1}}{i!} \right)
\]

→ There are 3 parts to this problem.

(i) In for loop, calculating numerator and denominator.

(ii) Finding sum in loop.

(iii) Multiplying \( x \) with the sum to get final sum.

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

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

// function to find factorial
int findfactorial(int p)
{
    int fact=1,i;
    for(i=1;i<=p;i++)
    {
        fact=fact*i;
    }
    return fact;
}

int main()
{
    int x,n,i;
    float sum;

    print("Enter the number x: \n");
    scanf("%d",x);
    print("Enter the number of terms n: \n");
    scanf("%d",n);

    for(i=1;i<=n;i++)
    {
        sum = sum + (float)(pow(-1,i+1))*(findfactorial(i));
    }
    sum = sum*x;

    print("The sum is: \n");
    printf("%f",sum);
    return 0;
}
```