Accenture Coding Questions with Answers

Q1. Binary String Operations

Problem statement:

The binary number system only uses two digits, 0 and 1. Any string that represents a number in the binary number system can be called a binary string. You are required to implement the following function:

int OperationsBinaryString(char *str);

The function accepts a string 'str' as its argument. The string 'str' consists of binary digits separated with an alphabet as follows:

- 'A' denotes AND operation
- 'B' denotes OR operation
- 'C' denotes XOR operation

You are required to calculate the result of the string 'str', scanning the string left to right, taking one operation at a time, and return the same.

Note:

- No order of priorities of operations is required.
- Length of 'str' is odd
- If 'str' is NULL or None(in case of python), return -1

Example:

Input:

str: ICOCICIAOBI

Output:

1

Explanation:

The alphabet in 'str' when expanded becomes "1 XOR 0 XOR 1 XOR 1 AND 0 OR 1", the result of the expression becomes 1, hence 1 is returned.

Code Solution in C++:

Sample Input:

str: 0C1A1B1C1C1B0A0

C++

```cpp
#include<iostream>

using namespace std;

int OperationsBinaryString(char* str)

{

    int len=0;

    int ans= str[0]-'0';

    for(len=0;str[len]!='\0';len++);

    for(int i=1;i<len-1;i+=2)

    {

        int j=i+1;

        if(str[i]=='A')

        {

            ans = ans & (str[j]-'0');

        }

        else if(str[i]=='B')

        {

            ans = ans | (str[j]-'0');

        }

        else if(str[i]=='C')

        {

            ans = ans ^ (str[j]-'0');

        }
```

```
                                                        23
    }
                                                        24
    return ans;
                                                        25
}
                                                        26
int main()
                                                        27
{
```

Output

```
0
```

Q2. Password Validation

<u>Problem statement</u>   You

are given a function:

int CheckPassword(char str[ ], int n);

The function accepts string 'std of size int as argument Implement the function which returns 1 if given string 'str' is a valid password else 0.

'str' is a valid password if it satisfies below conditions:

- At least 4 characters
- At least one numeric digit
- At least one Capital letter
- Must not have space or slash (/)
- Starting character must not be a number

Assumption: Input string will not be empty.
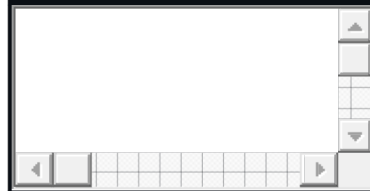

<u>Example:</u>

Input:

aA1_67

Output:

1

Code Solution in C++:

<u>Sample Input:</u> a987

abC012

C++

```cpp
#include<iostream>

using namespace std;

int CheckPassword(char str[], int n)

{

    bool isDigit=false,isCap=false,isSlashSpace=false,isNumStart=false,isLen=false;

    isLen = n>=4;

    isNumStart = (str[0]>='0' && str[0]<='9');


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

    {

        if(str[i]==' '|| str[i]=='/')

            isSlashSpace=true;

        else if(str[i]>='A'&&str[i]<='Z')

            isCap=true;

        else if(str[i]>='0' && str[i]<='9')

            isDigit=true;

    }

    if(!isNumStart && isDigit && isCap && !isSlashSpace && !isNumStart)

        return 1;

    else

        return 0;
```

```
                                                                    22
}
                                                                    23
int main()
                                                                    24
{
                                                                    25
    char str[100];
                                                                    26
    scanf("%[^\n]s",str);
                                                                    27
    int len=0;
```

Output

```
0
```

Q3. Find the maximum element and its index in an array

Problem statement:

You are given a function, void MaxInArray(int arr[], int length); The function accepts an integer array 'arr' of size 'length' as its argument. Implement the function to find the maximum element of the array and print the maximum element and its index to the standard output

(STDOUT). The maximum element and its index should be printed in separate lines.

Note:

- Array index starts with 0
- Maximum element and its index should be separated by a line in the output
- Assume there is only 1 maximum element in the array
- Print exactly what is asked, do not print any additional greeting messages

Example:

Input:

23 45 82 27 66 12 78 13 71 86

Output:

86

9

Explanation:

86 is the maximum element of the array at index 9.

Coding Solution in C++:

Sample Input

1 9 11 144 6 7 112 95

C++

```cpp
#include<iostream>

using namespace std;

void MaxInArray(int arr[],int length)

{

    int max=-999999,maxIdx=-1;

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

    {

        if(arr[i]>max)

        {

            max=arr[i];

            maxIdx=i;

        }

    }

    cout<<max<<endl<<maxIdx;

}

int main()

{

    int n;

    cin>>n;

    int a[n];

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

```
    cin>>a[i];
                                    23

  MaxInArray(a,n);
                                    24

}
```

Output
```
3
```

Q4. Replace Frequent Character

Problem statement

You are required to implement the following function: **char***

FrequentCharacterReplaced(char*' str, char x);

The function accepts a string 'str' and a character 'x' as its arguments. You are required to find the most frequent character in string 'str' and replace it with character 'x' across the string, and return the same.

Note:

If the frequency of two characters are the same, we have to consider the character with lower ascii value.

Example:

Input:

str: bbadbbababb

Output: ttadttatatt

Explanation:

The most frequent character in string 'str' is 'b', replacing 'b' with 't' will form string 'ttadttatatt', hence 'ttadttatatt' is returned.

Code Solution in C++:

Sample Input str:

jjkdkksjjdjf

```
C++
```

```cpp
#include<iostream>

using namespace std;

char* FrequentCharacterReplaced(char *str,char x)

{

    int ascii[26]={0};

    for(int i=0;str[i]!='\0';i++)

        ascii[str[i]-'a']++;

    int max=-99999;

    char maxchar;

    for(int i=0;i<26;i++)

    {

        if(ascii[i]>max)

        {

            max=ascii[i];

            maxchar=(char)'a'+i;

        }

    }

    for(int i=0;str[i]!='\0';i++)

    {

        if(str[i]==maxchar)

            str[i]=x;

    }
```

```
    return str;
                                                    24
}
                                                    25

int main()
                                                    26
{
                                                    27
    char a[100];
```

Output

```
yykdkksyydyf
```

Q5. Autobiographical Number

Problem statement

An Autobiographical Number is a number N such that the first digit of N represents the count of how many zeroes are there in N, the second digit represents the count of how many ones are there in N and so on.

You are given a function, **def FindAutoCount(n):**

The function accepts string "n" which is a number and checks whether the number is an autobiographical number or not. If it is, an integer is returned, i.e. the count of distinct numbers in 'n'. If not, it returns 0.

Assumption:

- The input string will not be longer than 10 characters.
- Input string will consist of numeric characters.

Note:

If string is None return 0.

Example:

Input: n:

"1210"

Output:

3

Explanation:

0th position in the input contains the number of 0 present in input, i.e. 1, in 1st position the count of number of 1s in input i.e. 2, in 2nd position the count of 2s in input i.e. 1, and in 3rd position the count of 3s i.e. 0, so the number is an autobiographical number.

Now unique numbers in the input are 0, 1, 2, so the count of unique numbers is 3. So 3 is returned.

Code solution in C++ Sample

Input:

n: "123"

```cpp
#include<iostream>

using namespace std;

int FindAutoCount(char *n)

{

    int valueCount[10]={0},freq[10]={0};

    int count=0;

    for(int i=0;n[i]!='\0';i++)

        valueCount[n[i]-'0']++;

    for(int i=0;i<10;i++)

        if(valueCount[i]>0)

            count++;

    for(int i=0;n[i]!='\0';i++)

        valueCount[i]=valueCount[i]-(n[i]-'0');

    for(int i=0;i<10;i++)

    {

        if(valueCount[i]!=0)

            return 0;
```

```
    }
    return count;
}
int main()
{
    char str[11];
    scanf("%s",str);
    cout<<FindAutoCount(str);
}
```

Output
```
0
```