**Output based Questions**

**1.)**
What will be the output of this code?

```
#include <stdio.h>
int fun ( char *s ) {
  if ( *s == '\0' ) return 0;
  if ( *s != *(s+1) ) {
    printf ("%c", *s);
    return 1 + fun (s+1);
  }
  return fun (s+1);
}
int main () {
  printf(" %d",fun("Yiipppeeee"));
  return 0;
}
```

Answer: Yipe 4

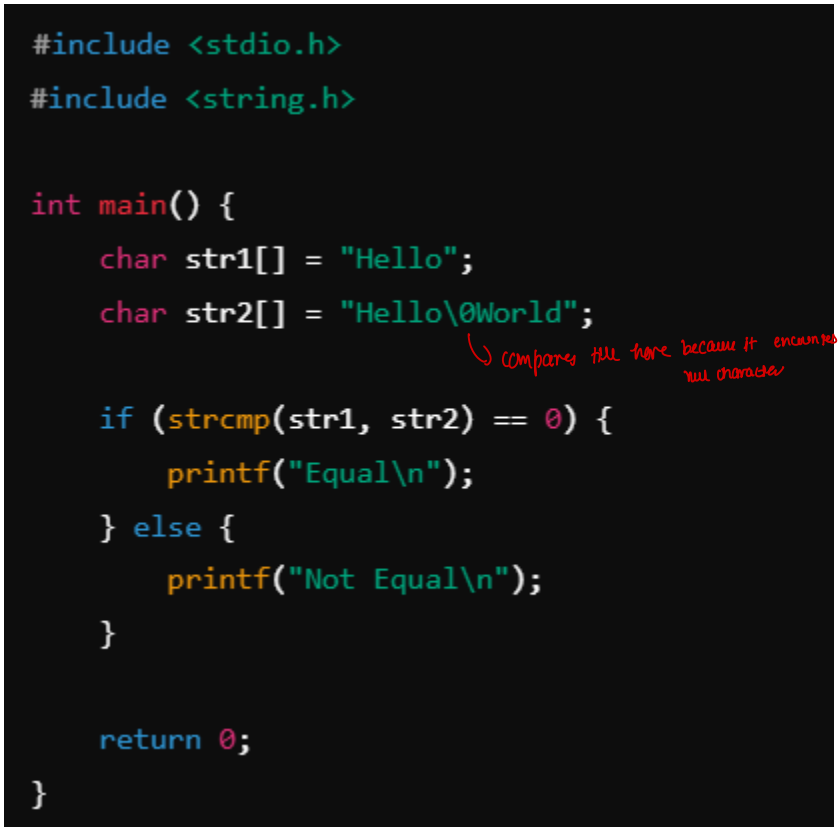
**2.)**

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

int main() {
    char s[12];
    char t[15];

    strcpy(s, "good");
    strcpy(t, "hi there");

    char *p = t + 4;
    char *temp = s;

    strcat(s, "bye");

    printf("s = %s\n", s);
    printf("t = %s\n", temp);
    printf("p = %s\n", p);

    return 0;
}
```

**What will be the output of the above program?**

**Ans:**
**s = goodbye**
**t = goodbye**
**p = here**

**3.)**
**What will be the output of the following code?**




**Ans: Equal**

**4.)**

```
#include <stdio.h>
int main () {
  char str[100];
  int d = 'a'-'A', i;
  for (i='A'; i<='Z'; ++i)
    str[i-'A'] = i;
  for (i=1; str[i]!='Z'; ++i)
    if ( (str[i] >= 'A') &&
         (str[i] <= 'Z') )
      str[i] += d;
  for (i=1; str[i]!='Z'; ++i)
    printf ("%c", str[i]);
  return 0;
}
```

**Ans: bcdefghijklmnopqrstuvwxy**

**5.) What will be displayed after the following C program fragment executes?**

```
char saying[] = "Too many cooks spoil the broth.";
char *p1, *p2;
p1 = saying;
p2 = saying + 8;
*p2 = '\0';
printf("%s\n", saying);
```

ANS: Too many

**6.)** Which of the following statements declares a string and initializes it to "work it"?

**A. char str[]="work it";**

B. char str[7]="work it";

C. string str="work it";

D. char str[8]="work it";

Ans : A

**7.)**

Which of the following statements prints the string "it" using the string `str` from above?

A. printf("%s", str+5);

B. printf("s", str+6);
C. printf("%s", str+7);
D. printf("%s", str+8);

**Ans: B**

**Note: Questions 8, 9, and 10 in the output-based questions seem to be more like fill-in-the-blank style questions about the code. Addressing those types of questions would help get a better understanding of the complete code implementation and the underlying logic. Having that level of clarity would be very helpful.** These 3 questions are inspired from Prof. Harish Karnick's CSD 101 papers.

**8.)**
The program fragment below contains the function pass
Check that is part of a password checking program. A password is considered valid if it satisfies the following rules:

1. It must have at least one upper case letter.
2. It must have at least one lower case letter.
3. It must have at least one digit.
4. It must have at least one of the symbols !, #, @.
5. It can be at most 8 characters long.

The function returns true if the password is valid and false if it is invalid.

(a) Fill in the parts shown by ??1?? to ??6?? so that the function works correctly. Each missing part is a single expression and/or one line of code.

```c
#include <stdio.h>
#include <ctype.h>
#define N 15

int passCheck(char paswd[], int n) {
    // n is the length of password
    int upper, lower, digit, special;
    char c;

    ??1??;

    for (int i=0; i<n; i++) {
        c=paswd[i];
        if (!upper && isupper(c)) ??2??;
        if (??3??) lower++;
        if (??4??) digit++;
        if (??5??) special++;
    }

    return (??6??);
}
```

**ANS:**
??1?? = upper=lower=digit=special=0;
??2?? = upper++; or equivalent
??3?? = !lower && islower(c) or equivalent
??4?? = !digit && isdigit(c) or equivalent
??5?? = !special && (c==(int)'@' || c==(int)'!' || c==(int)'#') or equivalent
??6?? = upper && lower && digit && special & n<=8 or equivalent

(b) What is the minimum length of a valid password?

**ANS:** Since the first 4 conditions of the specification must be satisfied the minimum length is 4.

**9.)**

Recall that in C characters are encoded as unsigned integers. The program fragment below reads upto 30 printable characters and changes the case of every alphabetic character (i.e. A-Z, a-z) from upper case to lower case and vice versa. Remember the codes of A-Z (and a-z) are sequential. Some parts of the program are replaced by ?1? to ?8?.

```
/*Reads a string of upto 30 printable characters and changes
the case of every alphabetic character from upper case to
lower case and vice versa.
*/
char str[?1?], c;
int i=0;
printf("Give the string (<=30 chars)=");
while((c=getchar())!=?2? && i<30) str[?3?]=c;
str[?4?]='\0';
for (i=0; (c=str[i])!=?5?; i++)
   if (c>='A' && c<='Z') str[i]+=?6?;
   else
      if (?7?) str[i]+=?8?;
printf("%s\n",str);
exit(0);
```

a) Fill in code for the replaced parts so that the program works correctly.
ANS:
**?1?** 31
**?2?** '\n'
**?3?** i++
**?4?** i
**?5?** '\0'
**?6?** ('a'-'A') or equivalent
**?7?** (c>='a' && c<='z') or equivalent ?8? ('A'-'a') or equivalent

b) What will be printed out by the code fragment in part (a) if the input is: "IsThisAStriNGOf25Chars?"?

ANS: "iStHISasTRIngoF25cHARS?"

**10.)**

Carefully study the code fragment given below and answer the following questions. Assume the fragment is a part of a program that compiles.

```
4  void swap(char *s1, char *s2) {
5    char *tmp;
6    tmp=s1;
7    s1=s2;
8    s2=tmp;
9  }
10 int main(void) {
11   char *str[]={"one","two","three","four"}, *tmp;
12   int n=4;
13   for(int i=0; i<n-1; i++)
14     if(strcmp(str[i],str[i+1])>0) {
15       tmp=str[i];
16       str[i]=str[i+1];
17       str[i+1]=tmp;
18     }
19   printf("%s, %s, %s, %s\n", str[0], str[1], str[2], str[3]);
20   return 0;
21 }
```

Note: The function `strcmp(s1, s2)` returns a value greater than 0 if string `s1` is later than `s2` in dictionary order and a value less than 0 if `s1` is earlier than `s2`.

a) What will be the output when the program is executed?
**ANS:** one, three, four, two
b) What will be the output if we replace lines 15 to 17 by the line `swap(str[i], str[i+1]);` and execute the program?
**ANS:** one, two, three, four

c) Will the outputs in (a) and (b) be the same or different? Justify your answer.

**ANS:** Different. Due to call by value parameter passing copies of the pointers are passed to swap. These pointer copies are exchanged in swap but the contents pointed to are not exchanged so swap does not change the contents of the array. The original pointers in the calling function remain unchanged.

**Programming questions**

**1.)**
**Question : Check Anagrams**

Write a function that checks if two given strings are anagrams. Two strings are considered anagrams if they contain the same characters in the same frequency but may have different orders. Ignore case differences.

**Input**:

- Two strings `str1` and `str2`, with alphabetic characters only.

**Output**:

- Print `1` if the strings are anagrams, otherwise print `0`.

Input: str1 = "listen", str2 = "silent"
Output: 1

Input: str1 = "hello", str2 = "world"
Output: 0

**Example:**

```
Input: str1 = "listen", str2 = "silent"
Output: 1

Input: str1 = "hello", str2 = "world"
Output: 0
```

**2.)**

**Question**: Compare Version Numbers
Write a function that compares two version numbers given as strings. Each version number consists of numbers separated by dots (e.g., "1.0.2"). Return `1` if the first version is greater, `-1` if the second is greater, and `0` if they are equal.

**Input**:

- Two version strings, `version1` and `version2`.

**Output**:

- An integer indicating if `version1` is greater, smaller, or equal to `version2`.

**Example**:

```
Input: "1.0.2", "1.0.10"
Output: -1

Input: "1.2.0", "1.1.9"
Output: 1

Input: "1.0.0", "1.0"
Output: 0
```

**3.)**

## Question: Isomorphic Strings

Given two strings `s` and `t`, determine if they are **isomorphic**.

Two strings are considered isomorphic if the characters in one string can be replaced to get the second string, such that:

- Each character in `s` can be mapped to exactly one character in `t`.
- No two characters in `s` map to the same character in `t` (i.e., one-to-one mapping).
- The order of characters must be preserved.

**You need to implement a function that checks if the two strings are isomorphic.**

**Function Signature : `bool areIsomorphic(char *s, char *t);`**

**Example:**
Input:
s = "egg"
t = "add"

Output:

Strings are isomorphic

**4.)**

**Question: Check if String is a Subsequence**

**Given two strings, `str1` and `str2`, check if `str1` is a subsequence of `str2`. A string is a subsequence if it appears in the same order but not necessarily consecutively.**

**Input:**

- **Two strings, `str1` (the potential subsequence) and `str2`.**

**Output:**

- **Print `Yes` if `str1` is a subsequence of `str2`, otherwise print `No`.**

**Example:**

```
Input: "abc", "aebdc"

Output: Yes

Input: "ace", "abcde"

Output: Yes

Input: "aec", "abcde"

Output: No
```

**5.)**

**Question: Remove Consecutive Duplicates**
**Given a string, remove consecutive duplicate characters and return the modified string.**

**Input:**

- **A single string `str`.**

**Output:**

- **A string with consecutive duplicates removed.**

**Example:**

```
Input: "aabbccdde"
Output: "abcde"

Input: "aaabbbccc"
Output: "abc"

Input: "abcd"
Output: "abcd"
```

**6.)**

**Question: Count and Replace Duplicates**
**Count the frequency of each character in a string and replace duplicates with `#`.**

**Input:**

- **A single string `str`.**

**Output:**

- **Print the modified string with duplicates replaced by `#`.**

**Example:**

```
Input: "programming"
Output: "prog#amm###"
```

**7.)**

**Question: Roman to Integer Conversion**
**Write a function that converts a Roman numeral to an integer. Follow the Roman numeral rules where:**

- **`I` is 1, `V` is 5, `X` is 10, `L` is 50, `C` is 100, `D` is 500, and `M` is 1000.**
- **If a smaller numeral is placed before a larger one (e.g., `IV`), it is subtracted.**

**Input:**

- **A string representing the Roman numeral (e.g., `XIV` or `MCMXCIV`).**

**Output:**

- **An integer representing the converted Roman numeral.**

**Example:**

```
Input: "XIV"
Output: 14

Input: "MCMXCIV"
Output: 1994
```

**8.)**

**Question: Group Anagrams**
**Given an array of strings, group the anagrams together. Anagrams contain the same characters in a different order.**

**Input:**

- **An array of strings `arr` and an integer `n` representing the number of strings.**

**Output:**

- **Print each group of anagrams.**

**Example:**

```
Input: ["listen", "silent", "enlist", "inlets", "stone", "notes"]
Output:
Group 1: listen, silent, enlist, inlets
Group 2: stone, notes
```

**9.)**

**Question: Valid Numbers**
**Determine if a given string can represent a valid integer or decimal number.**

**Input:**

- **A single string `str`.**

**Output:**

- **Print `Yes` if the string is a valid number, otherwise `No`.**

**Example:**

```
Input: "123.45"
Output: Yes

Input: "abc"
Output: No
```

**10.)**

**Question: String to Integer**
**Implement a function that converts a string to an integer.**

**Input:**

- **A single string `str`.**

**Output:**

- **An integer representing the converted value.**

**Example:**

```
Input: "123"
Output: 123

Input: "-456"
Output: -456
```

**11.)**

**Question:**
**Write a function to remove all spaces from a given string.**

**Input:**

- **A string `s`.**

**Output:**

- **The string without spaces.**

**Example:**

```
Input: "hello world"
Output: "helloworld"
```

**12.)**

**Question: Remove Spaces From Strings**
**Write a function to count the occurrences of a specific character in a given string.**

**Input:**

- **A string `s` and a character `ch`.**

**Output:**

- **The number of occurrences of the character `ch` in the string.**

**Example:**

```
Input: "hello", 'l'
Output: 2
```

**13.)**

**Question: Count Vowels in a String**
**Write a function to count the number of vowels in a given string.**

**Input:**

- **A string `s`.**

**Output:**

- **The number of vowels in the string.**

**Example:**

```
Input: "hello"
Output: 2
```

**14.)**

**Question: Find the First Non-Repeated Character in a String**
**Write a function that returns the first non-repeated character in a given string.**

**Input:**

- **A string `s`.**

**Output:**

- **The first non-repeated character, or `-1` if there is no non-repeated character.**

**Example:**

```
Input: "swiss"
Output: 'w'

Input: "aabbcc"
Output: -1
```

**15) Write a function to reverse the String using Recursion**
**16) Write a function to copy one string to another using Recursion**
**17) Write a function to find the first capital Letter in a string using Recursion**

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
void bubbleSort(char *str,int len)
{
    for(int i=0;i<len;i++)
    {
        str[i]=toupper(str[i]);
    }
    for(int i=0;i<len-1;i++)
    {
        for(int j=0;j<len-i-1;j++)
        {
            if(str[j]>str[j+1])
            {
                char temp=str[j];
                str[j]=str[j+1];
                str[j+1]=temp;
            }
        }
    }
}
int checkAnagrams(char* str1,char* str2)
{
    bubbleSort(str1,strlen(str1));
    bubbleSort(str2,strlen(str2));
    for(int i=0;i<strlen(str1);i++)
    {
        if(*(str1+i)!=*(str2+i))
        return 0;
    }
    return 1;

}
int main()
{
    char str1[50],str2[50];
printf("ENTER TWO STRINGS \n");
scanf("%s",str1);
scanf("%s",str2);
if(checkAnagrams(str1,str2))
{
    printf("TWO STRINGS ARE ANAGRAMS\n");
}
else
{
    printf("THEY ARE NOT ANAGRAMS\n");
}
return 0;
}
```

ompareversion.c > ...

```c
#include <stdio.h>

int compareVersions(const char *version1, const char *version2) {
    int num1, num2;
```

version.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
int convertToInteger(char *str)
{
    int num=0;
    char *ch=strtok(str,".");
    while(ch)
    {
        num=num*10+(*ch-'0');
        ch=strtok(NULL,".");
    }
    return num;
}
int main()
{
    char version1[50], version2[50];
    printf("Enter first version number: ");
    scanf("%s", version1);
    printf("Enter second version number: ");
    scanf("%s", version2);
    if(strlen(version1)!=strlen(version2))
    {
        if(strlen(version1)<strlen(version2))
        {
            while(strlen(version1)<strlen(version2))
            {
                strcat(version1,".0");
            }
        }
        else
        {
            while(strlen(version2)<strlen(version2))
            {
                strcat(version2,".0");
            }
        }
    printf("VERSION AFTER MODIFICATIONS ARE %s %s \n",version1,version2);
    int v1=convertToInteger(version1);
    int v2=convertToInteger(version2);
    if (v1>v2) {
        printf("Version 1 is greater.\n");
    } else if (v2<v1) {
        printf("Version 2 is greater.\n");
    } else {
        printf("Versions are equal.\n");
    }

return 0;
}
```

strtok returns pointer to that token.

somorphic.c > isIsomorphic(char *, char *)

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
bool isIsomorphic(char *str1 ,char* str2) // str1--> s str2--> t
```

> main()

```c
#include <stdio.h>
#include <stdbool.h>
bool isIsomorphic(char* s,char* t)
{
    for(int i=0;i<strlen(s);i++)
    {
        char c1=s[i];
        char c2=t[i];
        for(int j=i;j<strlen(s);j++)
        {
            if(s[j]==c1)
            {
                if(t[j]!=c2)
                return false;
            }
            if(s[j]==c2)
            {
                if(t[j]!=c1)
                return false;
            }
        }
    }
    return true;
}
int main()
{
char str1[50],str2[50];
printf("ENTER TWO STRINGS\n");
scanf("%s %s",str1,str2);
if(isIsomorphic(str1,str2))
{
    printf("THEY ARE ISOMORPHIC STRINGS\n");
}
else
{
    printf("THEY ARE NOT ISOMORPHIC STRINGS\n");
}
return 0;
}

  return 0;
  }
```

ovedup.c > removeConsecutiveDuplicates(char *)

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
char* removeConsecutiveDuplicates(char *str1)
{
    char *str2=(char*)malloc(strlen(str1)*sizeof(char)+1
    int i=0,j=0;
    for(i=0;i<strlen(str1);i++)
    {
        if(str1[i]==str1[i-1])
        continue;
        else
        str2[j++]=str1[i];
    }
    str2[j++]='\0';
    return str2;
}
int main()
{
char str[50];
printf("ENTER A STRING\n");
scanf("%s",str);
char *ans=removeConsecutiveDuplicates(str);
printf("%s\n",ans);
return 0;
}
```

countandremove.c > main()

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
int main()
{
char str1[50];
printf("ENTER A STRING\n");
scanf("%s",str1);
for(int i=0;i<strlen(str1);i++)
{
    int count=0;
    for(int j=i;j<strlen(str1);j++)
    {
    if(str1[i]==str1[j] && str1[i]!='#')
    {
    count++;
    if(j!=i)
    str1[j]='#';
    }
    }
    if(str1[i]!='#')
    printf("Count of %c is %d\n",str1[i],count);
}
printf("%s\n",str1);
return 0;
}
```

manToInteger.c > main()

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>;
int charToInt(char ch)
{
    switch (ch)
    {
    case 'I':
        return 1;
    case 'V':
        return 5;
    case 'X':
        return 10;
    case 'L':
        return 50;
    case 'C':
        return 100;
    case 'D':
        return 500;
    case 'M':
        return 1000;
    }
}
int romanToInteger(char *str)
{
    int ans = 0;
    for (int i = 0; i < strlen(str); i++)
    {
        int current = charToInt(str[i]);
        int next = (i == strlen(str)-1) ? 0 : charToInt(str[i + 1]);
        if(current<next)
        ans-=current;
        else if(current>next)
        ans+=current;
    }
    return ans;
}
int main()
{
    char str[20];
    printf("ENTER A ROMAN INTEGER\n");
    scanf("%s",str);
    int ans=romanToInteger(str);
    printf("%d\n",ans);
    return 0;
}
```

groupAnagrams.c > main()

```c
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
void bubbleSort(char *str)
{
    for(int i=0;i<strlen(str)-1;i++)
    {
        for(int j=0;j<strlen(str)-i-1;j++)
        {
            if(str[j]>str[j+1])
            {
                char temp=str[j];
                str[j]=str[j+1];
                str[j+1]=temp;
            }
        }
    }
}
bool checkAnagrams(char* str1,char* str2)
{
    char str1cpy[50+1],str2cpy[50+1];
    strcpy(str1cpy,str1);
    strcpy(str2cpy,str2);
    bubbleSort(str1cpy);
    bubbleSort(str2cpy);
  if(strcmp(str1cpy,str2cpy)==0)
  return true;
  else
  return false;
}
int main()
{
    int n;
    printf("ENTER THE NUMBER OF STRINGS\n");
    scanf("%d",&n);
    char str[n][50];
    bool visited[n];
    printf("ENTER THE STRINGS INTO THE ARRAY\n");
    for(int i=0;i<n;i++)
    {
        scanf("%s",str[i]);
    }
    int k=1;
    for(int i=0;i<n;i++)
    {
        if(!visited[i])
        printf("Group number %d \n",k++);
        else
        continue;
        for(int j=i;j<n;j++)
        {
            if(str[i][0]!='\0')
            {
            if(checkAnagrams(str[i],str[j]))
            {
            printf("%s  ",str[j]);
            visited[j]=true;
            }
        }
        }
        printf("\n");
    }
return 0;
}
```

validNumbers.c > main()

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
bool validNumber(char* str)
{
    for(int i=0;i<strlen(str);i++)
    {
        if(str[i]=='.' || (str[i]>='0' && str[i]<='9'))
        continue;
        else
        return false;
    }
    return true;
}
int main()
{
    char str[50];
printf("ENTER A STRING\n");
scanf("%s",str);
if(validNumber(str))
printf("IT IS A VALID NUMBER\n");
else
printf("IT IS NOT A VALID NUMBER\n");
return 0;
}
```

stringToInteger.c | countandremove | ex.c | strcat.c

stringToInteger.c > stringToInteger(char *)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
int stringToInteger(char *str)
{
    int sign=1;
    int number=0,i=0;
    if(str[i]=='-')
    {
    sign=-1;
    i++;
    }
    while(str[i]!='\0')
    {
        number=number*10+(str[i]-'0');
        i++;
    }
    return sign*number;
}
int main()
{
 char str[50];
 int num;
printf("ENTER A STRING\n");
scanf("%s",str);
num=stringToInteger(str);
printf("NUMBER IN INTEGER IS %d \n",num);
return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

stringToInteger.c | removeSpace.c | c.json | removedup

removeSpace.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
void removeSpaces(char* str)
{
    int j=0;
    for(int i=0;i<strlen(str);i++)
    {
        if(str[i]!=' ')
        str[j++]=str[i];
        else
        continue;
    }
    str[j]='\0';
}
int main()
{
    char str[50];
printf("ENTER A STRING\n");
scanf("%[^\n]s",str);
removeSpaces(str);
printf("STRING WITHOUT SPACES IS %s\n",str);
return 0;
}
```

countCh.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
int countCh(char *str,char ch)
{
    int i=0,count=0;
    while(str[i]!='\0')
    {
        if(str[i]==ch)
        count++;
        i++;
    }
    return count;
}
int main()
{
   char str[50];
   char ch;
printf("ENTER A STRING:  \n");
scanf("%[^\n]s",str);
printf("ENTER A CHARCATER : \n");
getchar();
scanf("%c",&ch);
int c=countCh(str,ch);
printf("STRING THE OCCURENCE OF %c in %s is %d\n",ch,str,c);
return 0;
}
```

nonrep.c > main()

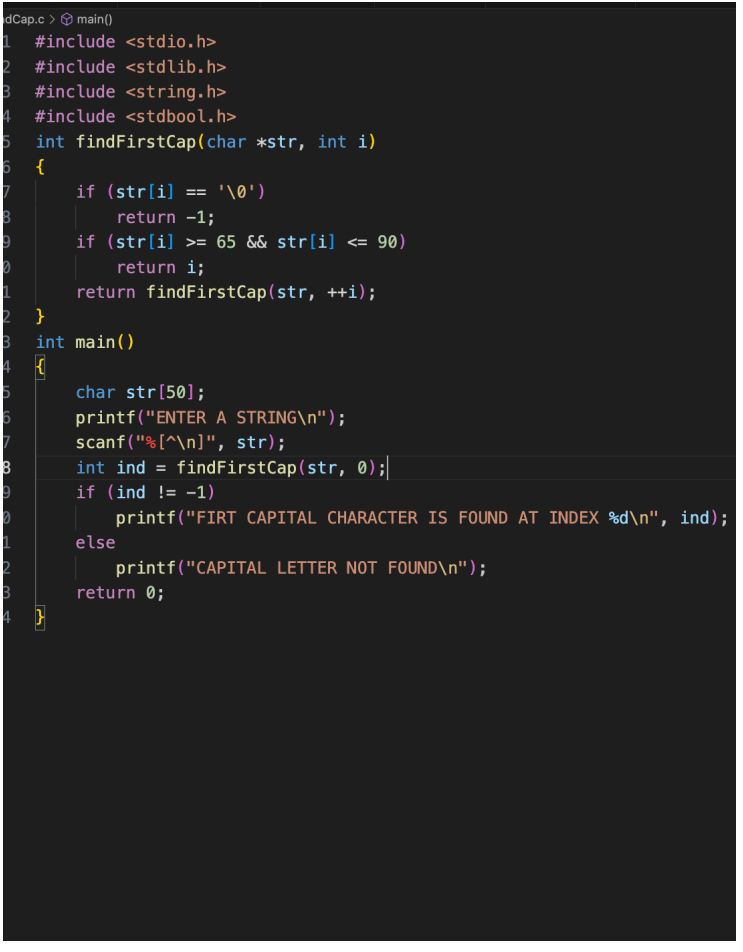
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
char nonRepeatedCharacter(char* str)
{
    int i=0,j=0;
    while(str[i]!='\0')
    {
        int count=0;
        j=i+1;
        while(str[j]!='\0')
        {
            if(str[i]==str[j])
            {
            count++;
            str[j]='\0';
            }
            j++;
        }
        if(count==0)
        return str[i];
        i++;
    }
    return '\0';
}
int main()
{
    char str[50];
printf("ENTER A STRING \n");
scanf("%s",str);
char ch=nonRepeatedCharacter(str);
if(ch!='\0')
printf("FIRST NON REPEATED CHARACTER IS %c\n",ch);
else
printf("-1");
return 0;
}
```

StringRecur.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
void reverseString(char* str,int start,int end)
{
   if(start>end)
   return;
   else{
    char temp=str[start];
    str[start]=str[end];
    str[end]=temp;
    reverseString(str,start+1,end-1);
   }
}
int main()
{
    char str[50];
printf("ENTER A STRING TO BE REVERSED\n");
scanf("%[^\n]s",str);
reverseString(str,0,strlen(str)-1);
printf("REVERSED STRING IS %s\n",str);
return 0;
}
```

opystrRecur.c > copystr(char *, char *, int)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
void copystr(char* dest,char* src,int i)
{
    if(src[i]!='\0')
    {
        dest[i]=src[i];
        copystr(dest,src,++i);
    }
    else
    {
        dest[i]='\0';
    return;
    }
}
int main()
{
    char src[50];
    char dest[50];
printf("ENTER A STRING\n");
scanf("%[^\n]s",src);
strcpy(dest,src);
printf("SOURCE STRING IS %s \n",src);
printf("DESTINATION STRING IS %s \n",dest);
return 0;
}
```


dCap.c > main()
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
int findFirstCap(char *str, int i)
{
if (str[i] == '\0')
return -1;
if (str[i] >= 65 && str[i] <= 90)
return i;
return findFirstCap(str, ++i);
}
int main()
{
char str[50];
printf("ENTER A STRING\n");
scanf("%[^\n]", str);
int ind = findFirstCap(str, 0);
if (ind != -1)
printf("FIRT CAPITAL CHARACTER IS FOUND AT INDEX %d\n", ind);
else
printf("CAPITAL LETTER NOT FOUND\n");
return 0;
}

**Theory Questions**

1.) How are strings represented in C's memory? Explain the null terminator character and its role in string manipulation
2.) Declare an array of three strings and initialize it (in the declaration statement itself) to contain the following three strings: cat, dog, giraffe.
**3.) How would you implement your own version of strcpy()?**
4.) What is the difference between the following two string declarations, and how might it affect string operations?

```
char str1[] = "Hello";
char str2[] = "Hello\0World";
```

5.) What is the difference between the following two string declarations, and how might it affect string operations?

```
char str1[6] = "Hello";
char str2[6] = {'H', 'e', 'l', 'l', 'o'};
```

6.) What happens if you try to use `strcat()` to append a string that is longer than the remaining space in the dest
char dest[10] = "Hello";
strcat(dest, ", world!");
7.) Describe the difference between statically and dynamically allocated strings.
8.) Explain different techniques for traversing a string, such as using pointers, indices,.
9.) What are common operations you can perform on strings?
10.) How can I compare two strings in C without using strcmp()
11.) What is the role of the null character (\0) in string manipulation in C?
12.) Difference between malloc(), calloc(), and realloc()