Qw WAP to input & display a string using scanf function.

using getchar & gets function -> getchar function used to single character from the terminal. We can use this function repeatedly to read successive single characters from the input & place them into a character array. The reading is terminated when the new line character (\n) is entered & the null character is then inserted at the end of the string the getchar function call takes the form.

```
char ch;
ch = getchar ( );
```

for ex:-

```
main ( )
{
char line [81], ch;
int c = 0;
printf ( " enter text @ ");
do
{
ch = getchar ( );
line [c] = ch;
c++;
}
while (ch != '\n');
line [c] = '\0';
printf ( " \n %s \n ", line)
}
```

(4)

gets function :- Another and more convenient method of reading a string of text containing white spaces is to use the library function gets available in the <stdio.h>

This is a simple function with one string parameter and called as under.

gets (str)

It reads character from the keyboard until a new line character is encountered and then append a null character to the string.

```
char line [80];
gets [line);
printf ( " %s ", line);
or
puts ( line);
```

Writing strings to function :-

using printf function ;

printf (" %s ", name);

Using Putchar & Puts functions :-

Putchar

```
char ch = 'A';
putchar (ch);
```

The function putchar requires one parameter this stmt is equivalent to

```
printf ("%c", ch);
```

Another & more convinient way of printing string values is to use the function puts. (this is a one parameter function & invoked as under -

```
puts (str);
```

for ex:-

```
char line [80];
gets (line);
puts (line);
```

Q WAP to input and display string using gets & puts.

Copy String without lib

C does not provides operators that work on strings directly. for instance we can not assign one string to another directly.

for ex:-

```
String2 = "ABC";
String1 = String2;
```

are not valid statements. if we really want to copy the characters in String2 into String1, we may do so on a character by character basis.

for ex:-

Q WAP to copy one string into another & count the number of characters copied.

```
Void main ( )
{
char String1[80], String2[80];
int i;
printf ("enter a String \n");
scanf ("%s", String2);
for (i=0; String2[i] != '\0'; i++)
{
String1[i] = String2[i];
}
```

```
String1[i] = '\0';
printf("\n");
printf("%s\n", String1);
printf("No. of characters = %d\n", i);
}
```

Copy Strings using String handling function:
The Strcpy function works almost like a String assignment operator,
It takes the form -
Strcpy (String1, String2);

```
#include <stdio.h>
void main()
{
char name[30];
Strcpy (name, "America");
printf("The name is %s\n", name);
getch();
}
```

Concatenation Without using library function we cannot assign one String to another directly, We cannot join two Strings together by the simple arithmetic addition.

String 3 = Str1 + Str2.
Str2 = Str1 + "hello";
are not valid.
the characters from Str1 & String 2 should be copied into the String 3 one after the other.

```
void main()
{
int i, j, k;
char fname[10] = {"Ram"};
char sname[10] = {"Pratap"};
char lname[10] = {"Singh"};
char name[30];

// Copy fname into name -
for (i=0; fname[i] != '\0'; i++)
name[i] = fname[i];   0 1 2 3 4 5 6 7 8 9 10

// End fname with space;
name[i] = ' ';

// Copy second name.
for (j=0; sname[j] != '\0'; j++)
name[i+j+1] = sname[j];

// End sname with space.
```

Pointers & Strings :-

```
Char Str[ ] = "Hello";
Char *p = "Hello";
```

We cannot assign a string to another, where as we can assign a char pointer to another char pointer.

for ex :-

```
Void main ( )
{
Char Str1 [ ] = "Hello";
Char Str2 [10];
Char *S = "Good Morning";
Char *q;
Str2 = Str1;   // error
q = S;   // Works
}
```

Also, once a string has been defined, it cannot be initialized to another set of characters.

for ex :-

```
Void main ( )
{
Char Str1 [ ] = "Hello";
Char *p = "Hello";
Str1 = "Bye";   // error.
p = "Bye";   // Works
}
```

Q. WAP to cost count the length of string using user defined function.

```
int xStrlen (Char *s)
{
int l = 0;
While (*s != '\0')
{
l++;
s++
}
return (l);
}
Void main ( )
{
Char Str [20];
int len1, len2;
puts ("Enter a String");
gets (Str);
len1 = xStrlen (Str);
len2 = xStrlen ("Good morning");
printf ("String = %s length = %d \n",
Str, len1);
printf ("String = %s length = %d \n",
"Good morning", len+);
}
```

// Copy function -

```
void x strcpy (char *t, char *s)
{
while (*s1 = '\0')
{
*t = *s;
s++;
t++;
}
*t = '\0';
}
```

Strcat or strcmp do it yourself

Two-d- array of characters

WAP to input & display 2-d- array of characters

WAP to enter a string & count the char, values, words.

imp // Program to count words, vowels, cites, space digits, consonant & others.

```
void main ()
{
char a[80], c;
int i, v=0, co=0, ch=0, w=0, sp=0,
digit=0;
int others = 0;
printf ("enter a line");
gets (a);
for (i=0; a[i] != '\0'; i++)
{
ch++;
c = to upper (a[i]);
if (c=='A' || c=='E' || c=='I' ||
c=='O' || c=='U')
++v;
else
if (c>='A' && c<='Z')
++co;
else if (c>='0' && c<='9')
++digit;
else if (c==' ' || c=='\t')
{
++sp;
++w;
}
++others;
}
printf ("
")
}
```

Dynamic memory allocation :- C language requires the no. of elements in an array to be specified at compile time But it may not be able to do so always. Our initial judgement of size, if it

is wrong may cause failure of the program or wastage of memory space. The process of allocating memory at run time is known as dynamic memory allocation. There are four library routines known as memory management function that can be used for allocating & freeing memory during program executions.

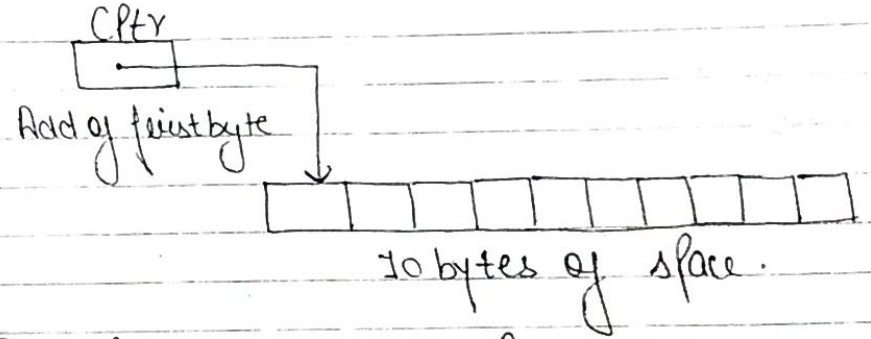
Function name	Task
malloc	Allocates request size of bytes and returns a pointer to the first byte of the allocated space.
calloc	Allocates space for an array of elements, initializes them to zero & then returns a pointer to the memory.
free	Frees previously allocated space
realloc	Modifies the size of previously allocated space.

1. Allocating a Block of Memory MALLOC:- A block of memory may be allocated using the function malloc. The malloc function reserves a block of memory of specified size & returns a pointer of type void. This means that we can assign it to any type of pointer. It takes the following form:-

```
ptr = (cast-type*) malloc (byte-size);
```

For ex:- `x = (int*) malloc (100*sizeof(int));`

OR `(ptr = (char*) malloc (10);`



2. Allocating Multiple Blocks of memory - calloc. Calloc is used for requesting memory space at run time for storing derived data types. Such as arrays & Structures. While malloc allocates a single block of storage space, calloc allocates

Multiple blocks of Storage, each of the same size, and then sets all bytes to zero.

ptr = (cast-type*) calloc (n, element-size)

(3) Releasing the used space - free:-
With the dynamic run-time allocation, it is our responsibility to release the space when it is not required.

free (ptr);

Altering the Size of a Block - Realloc:-
The previously allocated memory is not sufficient and we need additional space for more elements. It is also possible that the memory allocated is much larger than necessary & we want to change it. In both the cases, we can change the memory size already allocated with the help of realloc function.

for ex:- Original allocation is ptr = malloc (size);
then reallocation of space may be done by the stmt.
ptr = realloc(ptr, new size);

Q WAP.

```
#include <calloc.h>
{ int *p, n, i;
printf("Enter the value of n");
scanf("%d", &n);
p=(int*) malloc (n* sizeof(int));
if (p==NULL)
{
printf("Unable");
exit(0);
}
else
{
printf("Input value of array");
for(i=0 to n)
{
scanf("%d", (p+i));
}
printf("value of dynamic array\n");
for(i=0 to 0-1)
printf("%d", *(p+i));
}
free(p);
}
```

strupr, strlwr } - string.h { to upper, lowercase

Page no. 346