| S.No: 28 | Exp. Name: ***Design a C program that sorts the strings using array of pointers.*** | Date:2023-04-18 |
|---|---|---|

**Aim:**

Design a C program that sorts the strings using array of pointers.

**Sample input output**

```
Sample input-output -1:

Enter the number of strings: 2
Enter string 1: Tantra
Enter string 2: Code
Before Sorting
Tantra
Code
After Sorting
Code
Tantra
Sample input-output -2:

Enter the number of strings: 3
Enter string 1: India
Enter string 2: USA
Enter string 3: Japan
Before Sorting
India
USA
Japan
After Sorting
India
Japan
USA
```

**Source Code:**

stringssort.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void sort(char *s[],int);
void main()
{
   char*temp;
   int i,j,diff,num_strings;
   char*strarray[10];
   printf("Enter the number of strings: ");
   scanf("%d",&num_strings);
   if(num_strings>10)
   {
      printf("Sorry,maximum strings allowed is 10. Defaulting.");
      num_strings =10;
   }
   for(i=0;i<num_strings;i++)
```

```
   {
      printf("Enter string %d: ",i+1);
      strarray[i] =(char*)malloc(10 *sizeof(char));
      scanf("%s",strarray[i]);
   }
   printf("Before Sorting\n");
   for(i=0;i<num_strings;i++)
   {
      printf("%s\n",strarray[i]);
   }
   sort(strarray,num_strings);
   printf("After Sorting\n");
   for(i=0;i<num_strings;i++)
   {
      printf("%s\n",strarray[i]);
   }
}
void sort(char *s[],int num_strings)
{
   char*temp;
   int item,i;
   for(item=0;item<num_strings;item++)
   {
      temp =s[item];
      for(i=item;i>0&&strcasecmp(s[i-1],temp)>0;i--);
      {
         memmove(&s[i+1],&s[i],(item-i)*sizeof(char *));
         s[i]=temp;
      }
   }
}
```

## Execution Results - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| Enter the number of strings: 2 |
| Enter string 1: Tantra |
| Enter string 2: Code |
| Before Sorting |
| Tantra |
| Code |
| After Sorting |
| Code |
| Tantra |

| Test Case - 2 |
|---|
| User Output |
| Enter the number of strings: 3 |
| Enter string 1: Dhoni |
| Enter string 2: Kohli |
| Enter string 3: Rohit |
| Before Sorting |
| Dhoni |
| Kohli |
| Rohit |
| After Sorting |
| Dhoni |
| Kohli |
| Rohit |


ID: 224G1A0557
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

| S.No: 27 | Exp. Name: ***Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.*** | Date:2023-04-01 |
|---|---|---|





**Aim:**

Develop a C program which takes two numbers as command line arguments and finds all the common factors of those two numbers.

**Sample input output**

```
Sample input output -1:

Cmd Args : 10 20
Common factors for 10 and 20 are: 1     2       5       10

Sample input output -2:

Cmd Args : 45 23
Common factors for 45 and 23 are: 1
```

**Note:** Do use the printf() function with a newline character (\n) at the end.

**Source Code:**

common_factors.c

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char*argv[])
{
   int a,b;
   int i, small;
   a=atoi(argv[1]);
   b=atoi(argv[2]);
   small=(a<b)?a:b;
   printf("Common factors for %d and %d are: ",a,b);
   for(i=1;i<=small;i++)
   {
      if(a%i==0&&b%i==0)
      printf("%d\t",i);
   }
   printf("\n");
   return 0;
}
```

## Execution Results - All test cases have succeeded!

| Test Case - 1 |
| --- |
| User Output |
| Common factors for 10 and 20 are: 1 2 5 10 |

| Test Case - 2 |
| --- |
| User Output |
| Common factors for 18 and 39 are: 1 3 |

| S.No: 26 | Exp. Name: ***Illustrate the use of extern variables*** | Date:2023-04-02 |
|---|---|---|

**Aim:**

Illustrate the use of extern variables.

Follow the instructions given in the comment lines to write code and the working of the extern variables.

**Source Code:**

main.c

```
// Use the variable initialized in extrafile.c
#include"extrafile.c"
void main() {
   printf("Value of the external integer is = %d\n", i);

#include <stdio.h>
int i=51;
}
```

extrafile.c

```
#include <stdio.h>
int i=51;
```

**Execution Results** - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| Value of the external integer is = 51 |



ID: 224G1A0557



Srinivasa Ramanujan Institute of Technology

| S.No: 25 | Exp. Name: ***Illustrate the use of register variables*** | Date:2023-03-31 |
|---|---|---|





**Aim:**

Illustrate the use of register variables.

- You can use the **register** storage class when you want to store local variables within functions or blocks in CPU registers instead of RAM to have quick access to these variables. For example, "counters" are a good candidate to be stored in the register.
- The keyword **register** is used to declare a register storage class. The variables declared using register storage class has lifespan throughout the program.
- It is similar to the auto storage class. The variable is limited to the particular block. The only difference is that the variables declared using register storage class are stored inside CPU registers instead of a memory. Register has faster access than that of the main memory.
- The variables declared using register storage class has no default value. These variables are often declared at the beginning of a program.
- Accessing the address of the register variables results in an error.

**Try it out**

```
A statement like
int *ptr = &weight;
will result in an error like
int *ptr = &weight;
address of register variable 'weight' requested
```

Follow the instructions given in the comment lines to understand the working of register variables.

**Source Code:**

register.c

```
#include <stdio.h>
void main() {
register int weight; // Declare a register variable weight of type int.
   printf("The default weight value is: %d\n",weight);
   weight=65;
   printf("The default weight value is: %d\n",weight);
   // Add the line described above to obtain the error.
}
```

**Execution Results** - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| The default weight value is: 1024482696 |

| S.No: 24 | Exp. Name: ***Illustrate the use of static variables*** | Date:2023-04-03 |
|---|---|---|





**Aim:**

Illustrate the use of static variables
The **static** variables are used within function/ file as local static variables.
They can also be used as a global variable
Static local variable is a local variable that retains and stores its value between function calls or block and remains visible only to the function or block in which it is defined.
Static global variables are global variables visible only to the file in which it is declared.
Static variable has a default initial value zero and is initialized only once in its lifetime.
Follow the instructions given in the comment lines to declare and initialize the static variables and understand the working of static variables.

**Source Code:**

static.c

```
#include <stdio.h>
void next(void);
static int counter=5;// Declare a global static variable 'counter' with an ini
tial value of 5.
void main() {
   while(counter<10) {
      next();
      counter++;
   }
   return 0;
}
void next( void ) {
   static iteration=10;// Declare a static integer variable 'iteration' with a
n initial value 10
   iteration ++;
   printf("iteration=%d and counter= %d\n", iteration, counter);
}
```

**Execution Results** - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| iteration=11 and counter= 5 |
| iteration=12 and counter= 6 |
| iteration=13 and counter= 7 |
| iteration=14 and counter= 8 |
| iteration=15 and counter= 9 |

| S.No: 23 | Exp. Name: ***Illustrate the use of auto variable*** | Date:2023-04-01 |
|---|---|---|

**Aim:**

Illustrate the use of auto variable.
The variables defined using **auto** storage class are called as local variables.
Auto stands for **automatic** storage class. A variable is in auto storage class by default if it is not explicitly specified.
The scope of an auto variable is **limited with the particular block only.**
Once the control goes out of the block, the access is destroyed. This means only the block in which the auto variable is declared can access it.
A keyword **auto** is used to define an auto storage class. By default, an auto variable contains a **garbage value**.
Follow the instructions given in the comment lines to declare auto variables and print their values at different places in the program.

**Source Code:**

auto.c

```
#include<stdio.h>
void main() {
auto int d=10;// Declare an auto variable d of type integer.
   printf("d=%d\n",d);// Print the value of d.
   {
      auto int d=4; // Declare and initialize the auto variable d with 4.
      {
         auto int d=6;// Declare and initialize the auto variable d with 6/
      printf("d=%d\n",d);  // Print the value of d.
      }
   printf("d=%d\n",d);  // Print the value of d.
   }
}
```

**Execution Results** - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| 32767 |
| 6 |
| 4 |

| S.No: 22 | Exp. Name: ***Write a C program to sort given elements using quick sort*** | Date:2023-04-02 |
|---|---|---|





**Aim:**

Write a program to sort (Ascending order) the given elements using quick sort technique.

**Note: Pick the first element as pivot. You will not be awarded marks if you do not follow this instruction.**

At the time of execution, the program should print the message on the console as:

```
Enter array size :
```

For example, if the user gives the input as:

```
Enter array size : 5
```

Next, the program should print the following message on the console as:

```
Enter 5 elements :
```

if the user gives the input as:

```
Enter 5 elements : 34 67 12 45 22
```

then the program should print the result as:

```
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67
```

Note: Do use the **printf()** function with a newline character **(\n).**

**Source Code:**

quicksortmain.c

```
#include<stdio.h>
void read(int [],int);
void display(int [],int);
void main()
{
   int arr[15],i,n;
   printf("Enter array size : ");
   scanf("%d",&n);
   printf("Enter %d elements : ",n);
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
   printf("Before sorting the elements are : ");
   display(arr,n);
   quickSort(arr,0,n-1);
   printf("After sorting the elements are : ");
   display(arr,n);
}
int i,j,temp,pivolt,left,right;
void display(int a[15],int n)
{
   for(i=0;i<n;i++)
```

```
   printf("%d ",a[i]);
   printf("\n");
}
int partition(int arr[15],int lb,int ub)
{
}
void quickSort(int a[15],int low,int high)
{
   left=low;
   right=high;
   pivolt=a[(low+high)/2];
   do
   {
      while(a[left]<pivolt)
      left++;
      while(a[right]>pivolt)
      right--;
      if(left<=right)
      {
         temp=a[left];
         a[left]=a[right];
         a[right]=temp;
         right--;
         left++;
      }
   }
   while(left<=right);
   if(low<right)
   quickSort(a,low,right);
   if(left<high)
   quickSort(a,left,high);
}
```



ID: 224G1A0557

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

quicksortfunctions.c

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

## Execution Results - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| Enter array size : 5 |
| Enter 5 elements : 34 67 12 45 22 |
| Before sorting the elements are : 34 67 12 45 22 |
| After sorting the elements are : 12 22 34 45 67 |

| Test Case - 2 |
|---|
| User Output |
| Enter array size : 8 |
| Enter 8 elements : 77 55 22 44 99 33 11 66 |
| Before sorting the elements are : 77 55 22 44 99 33 11 66 |
| After sorting the elements are : 11 22 33 44 55 66 77 99 |

| Test Case - 3 |
|---|
| User Output |
| Enter array size : 5 |
| Enter 5 elements : -32 -45 -67 -46 -14 |
| Before sorting the elements are : -32 -45 -67 -46 -14 |
| After sorting the elements are : -67 -46 -45 -32 -14 |


ID: 224G1A0557
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

| S.No: 21 | Exp. Name: ***Write a C program to sort the elements using selection sort - largest element method technique*** | Date:2023-04-02 |
|---|---|---|





**Aim:**

Write a program to sort the given array elements using selection sort largest element method.

At the time of execution, the program should print the message on the console as:

```
Enter value of n :
```

For example, if the user gives the input as:

```
Enter value of n : 3
```

Next, the program should print the messages one by one on the console as:

```
Enter element for a[0] :
Enter element for a[1] :
Enter element for a[2] :
```

if the user gives the input as:

```
Enter element for a[0] : 22
Enter element for a[1] : 33
Enter element for a[2] : 12
```

then the program should print the result as:

```
Before sorting the elements in the array are
Value of a[0] = 22
Value of a[1] = 33
Value of a[2] = 12
After sorting the elements in the array are
Value of a[0] = 12
Value of a[1] = 22
Value of a[2] = 33
```

Fill in the missing code so that it produces the desired result.

**Source Code:**

array.c

```
#include<stdio.h>
void main()
{
   int a[20],i,n,j,max,temp=0;
   printf("Enter value of n : ");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("Enter element for a[%d] : ",i);
      scanf("%d",&a[i]);
   }
   printf("Before sorting the elements in the array are\n");
   for(i=0;i<n;i++)
   printf("Value of a[%d] = %d\n",i,a[i]);
```

```
   for(i=n-1;i>0;i--)
   {
      max=1;
      for(j=i;j>=0;j--)
      {
         if(a[j]>=a[max])
         max=j;
      }
      temp=a[i];
      a[i]=a[max];
      a[max]=temp;
   }
   printf("After sorting the elements in the array are\n");
   for(i=0;i<n;i++)
   {
      printf("Value of a[%d] = %d\n",i,a[i]);
   }
}
```



ID: 224G1A0557

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

## Execution Results - All test cases have succeeded!

| Test Case - 1 |
| --- |
| User Output |
| Enter value of n : 3 |
| Enter element for a[0] : 15 68 48 |
| Enter element for a[1] : Enter element for a[2] : Before sorting the elements in th |
| Value of a[0] = 15 |
| Value of a[1] = 68 |
| Value of a[2] = 48 |
| After sorting the elements in the array are |
| Value of a[0] = 15 |
| Value of a[1] = 48 |
| Value of a[2] = 68 |



ID: 224G1A0557

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology