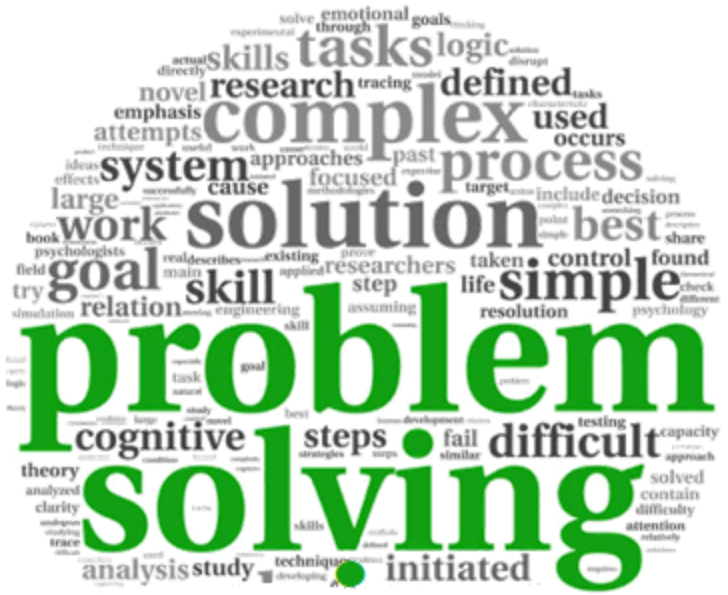


CSE 1051

20.2 FURTHER PROGRAMS ON RECURSION

Objectives:

To learn and understand the following concepts:

- ✓ To understand recursive algorithm
- ✓ To solve few problems including sorting using recursion

Session outcome:

At the end of session one will be able to :

- Understand recursion
- Write programs using recursive functions

Extra Problem- Sum of natural numbers

```
#include <stdio.h>
int sum(int n);

int main()
{
   int number, result;

   printf("Enter a positive integer: ");
   scanf("%d", &number);

   result = sum(number);

   printf("sum=%d", result);
}
```

```
int sum(int num)
{
   if (num!=0)
      return num + sum(num-1);
   else
      return num;
}
```

Output:

Enter a positive integer: 10

Sum= 55

Extra Problem- To count number of digits

```
#include <stdio.h>
int countDigits(int);
int main()
{
   int number;
   int count=0;

   printf("Enter a positive integer number: ");
   scanf("%d",&number);

   count=countDigits(number);

   printf("Number of digits is: %d\n",count);

   return 0;
}
```

```
int countDigits(int num)
{
   static int count=0;

   if(num>0)
   {
      count++;
      countDigits(num/10);
   }
   else
   {
      return count;
   }
}
```

Output:

```
Enter a positive integer number:  123
Number of digits is: 3
```

Extra Problem- To find sum of all digits

```
#include <stdio.h>
int sumDigits(int num);
int main()
{
   int number,sum;

   printf("Enter a positive integer number:
");
   scanf("%d",&number);

   sum=sumDigits(number);

   printf("Sum of all digits are: %d\n",sum);

   return 0;
}
```

```
int sumDigits(int num)
{
   static int sum=0;
   if(num>0)
   {
      sum+=(num%10);
      sumDigits(num/10);
   }
   else
   {
      return sum;
   }
}
```

Output:

```
Enter a positive integer number:  123
Sum of all digits are: 6
```

Extra Problem- Calculating power of a number

```
#include <stdio.h>
int power(int n1, int n2);

int main()
{
   int base, powerRaised, result;

   printf("Enter base number: ");
   scanf("%d",&base);

   printf("Enter power number);
   scanf("%d",&powerRaised);

   result = power(base, powerRaised);

   printf("%d^%d = %d", base, powerRaised, result);
   return 0;
}
```

```
int power(int base, int powerRaised)
{
   if (powerRaised != 0)
      return (base*power(base, powerRaised-1));
   else
      return 1;
}
```

Output:

Enter base number:3
Enter power number: 4
3 ^ 4=81

Extra Problem- To find length of a string

```
#include <stdio.h>
int length(char [], int);

int main()
{
   char str[20];
   int count;

   printf("Enter any string :: ");
   scanf("%s", str);
   count = length(str, 0);
   printf("The length of string=%d.\n",count);
   return 0;
}
```

```
int length(char str[], int index)
{
   if (str[index] == '\0')
   {
      return 0;
   }
   return (1 + length(str, index + 1));
}
```

Output:

Enter any string :: Manipal

The length of string= 7

Extra Problem-Binary Search

```
#include<stdio.h>
int binarySearch(int x[],int element,int start,int end);
int main(){
   int x[20],n,i,index,start=0,end,element;
   printf("Enter number of  elements: ");
   scanf("%d",&n);
   end = n;
   printf("Enter array elements: ");
   for(i=0;i<n;i++){
                scanf("%d",&x[i]);
   }
   printf("Enter the element to search: ");
   scanf("%d",&element);
   index = binarySearch(x,element,start,end-1);
   if(index == -1)
                printf("Element Not Found.\n");
   else
                printf("Element found at index : %d\n",index);
   return 0;
}
```

Extra Problem-Binary Search

```
int binarySearch(int x[],int element,int start,int end){
   int mid,noOfElements,i;
   mid = (int)(start+end)/2;
   if(start > end)
            return -1;
   if(x[mid] == element)
            return mid;
   else if(x[mid] < element){
            start = mid+1;
            return binarySearch(x,element,start,end);
   }
   else{
            end = mid-1;
            return binarySearch(x,element,start,end);
   }

}
```

Output:

```
Enter number of  elements: 5
Enter array elements: 1 2 3 4 5
Enter the element to search: 3
Element found at index : 2
```

Extra Problem- Recursive Sorting

Base Case:

if length of the list (n) = 1

No sorting, return

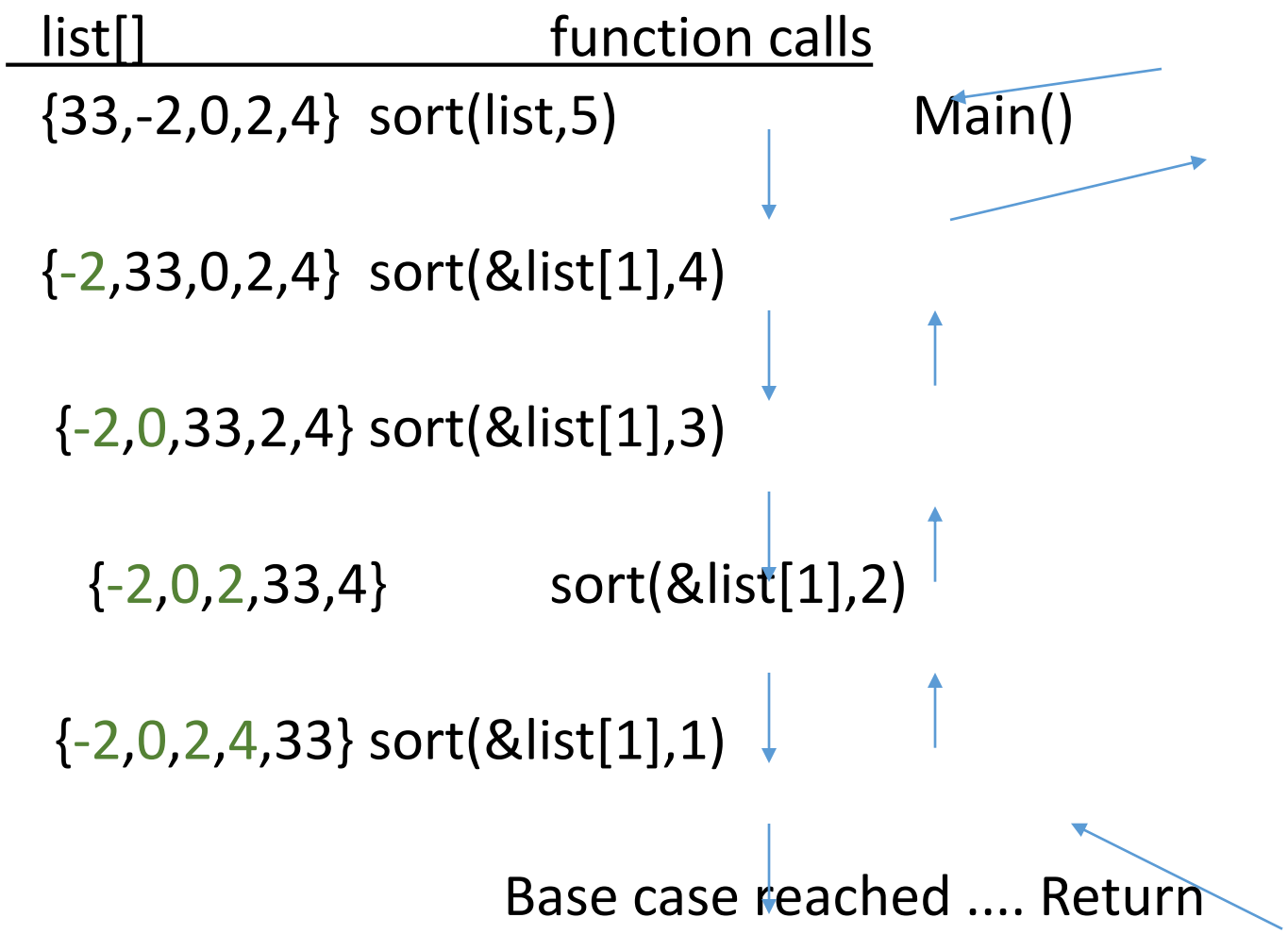
Recursive Call:

1. Find the smallest element in the list and place it in the 0^{th} position
2. Sort the unsorted array from 1.. n-1

sortR(&list[1], n-1)

For eg: list [ ] = {33,-2,0,2,4} n=5

Extra problem-Sorting

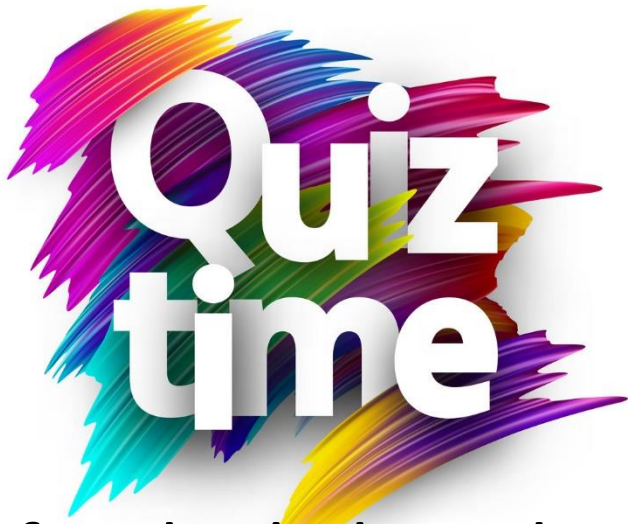


Extra problem - Sorting

```
sortR(list, n);// call of fn & display of sorted array in main()
```

```
int sortR(int list[], int ln){
int i, tmp, min;
if (ln == 1)
   return 0;
/* find index of smallest no */
min = 0;
for(i = 1; i < ln; i++)
  if (list[i] < list[min])
     min = i;
/* move smallest element to 0-th
     element */
tmp = list[0];
list[0] = list[min];
list[min] = tmp;
/* recursive call */
return sortR(&list[1], ln-1);
}
```

```
Output:
Orign. array-:  33  -2  0  2  4
Sorted array -: -2  0  2  4  33
```



Go to posts/chat box for the link to the question **PQn. S20.2**

submit your solution in next 2 minutes

The session will resume in 3 minutes

Summary

- Definition
- Recursive functions
- Problems Solving Using Recursion