**LAB #1: Write a program to compute an using recursion**

#include<stdio.h>

#include<conio.h>

int power(int x, int n);

int main()

{

intx,n;

printf("Enter value of x and n: \n");

scanf("%d%d",&x,&n);

printf("%d^%d = %d",x,n,power(x,n));

getch();

return 0;

}

int power(intx,int n)

{

if(n==0)

return 1;

else

return x\*power(x,n-1);

}

**LAB#2 Write a program to implement binary search using recursion**

#include<stdio.h>

#include<conio.h>

intbinSearch(intarr[],intkey,intl,int r);

int main()

{

intarr[] = {11,12,13,14,15,16,17,18,19,20};

int key=11;

intpos = binSearch(arr,key,0,10);

if(pos<0)

printf("Key item not found");

else

printf("Key item found at location %d\n",pos);

getch();

return 0;

}

intbinSearch(intarr[],int key, intl,int r)

{

if(l<r)

{

int m = (l+r)/2;

if(arr[m]==key)

return m+1;

else if(key<arr[m])

returnbinSearch(arr,key,l,m-1);

else

returnbinSearch(arr,key,m+1,r);

}

}

**LAB#3: Write a program to generate permutations and combinations**

//Program to generate permutation

#include<conio.h>

#include<stdio.h>

int factorial(int n);

int main()

{

intn,r,i;

printf("Enter value of n and r\n");

scanf("%d%d",&n,&r);

printf("The permutation P(%d, %d) = %d\n",n,r,factorial(n)/(factorial(n-r)));

getch();

return 0;

}

int factorial(int n)

{

if(n<=1)

return 1;

else

return n\*factorial(n-1);

}

**Output:**

Enter value of n and r

10 5

The permutation P(10, 5) = 30240

//Program to implement Combination

#include<conio.h>

#include<stdio.h>

int factorial(int n);

int main()

{

intn,r,i;

printf("Enter value of n and r\n");

scanf("%d%d",&n,&r);

printf("The combination C(%d, %d) = %d\n",n,r,factorial(n)/(factorial(n-r)\*factorial(r)));

getch();

return 0;

}

int factorial(int n)

{

if(n<=1)

return 1;

else

return n\*factorial(n-1);

}

**Output**

Enter value of n and r

12 10

The combination C(12, 10) = 66

**LAB#4: Write a C programs to implement quick sort**

#include<stdio.h>

#include<conio.h>

#include<math.h>

voidquickSort(intarr[],intl,int r);

int partition(intarr[],intl,int r);

int main()

{

intarr[] = {1,0,10,8,3,6,4,7,2,5};

inti;

printf("Array before sorting\n");

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

{

printf("%d\t",arr[i]);

}

quickSort(arr,0,9);

printf("\nArray after sorting\n");

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

{

printf("%d\t",arr[i]);

}

getch();

return 0;

}

voidquickSort(intarr[],intl,int r)

{

if(l<r){

int p = partition(arr,l,r);

quickSort(arr,l,p-1);

quickSort(arr,p+1,r);

}

}

int partition(intarr[],intl,int r)

{

intx,y,p,t;

x=l;

y=r;

p=arr[l];

while(x<y)

{

while(arr[x]<=p && x<=y)

x++;

while(arr[y]>p&& x<=y)

y--;

if(x<y)

{

If() t = arr[x];

arr[x] = arr[y];

arr[y]=t;

}

}

t = arr[l];

arr[l]=arr[y];

arr[y]=t;

return y;

}

**LAB#5: Write a C program to implement quick sort using randomized algorithm**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

intRandpartition(intarr[],intl,int r);

voidquickSort(intarr[],intl,int r);

int partition(intarr[],intl,int r);

int main()

{

intarr[] = {1,0,10,8,3,6,4,7,2,5};

inti;

printf("Array before sorting\n");

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

{

printf("%d\t",arr[i]);

}

quickSort(arr,0,9);

printf("\nArray after sorting\n");

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

{

printf("%d\t",arr[i]);

}

getch();

return 0;

}

voidquickSort(intarr[],intl,int r)

{

if(l<r){

int p = Randpartition(arr,l,r);

quickSort(arr,l,p-1);

quickSort(arr,p+1,r);

}

}

intRandpartition(intarr[],intl,int r)

{

intrp = l+rand()%(r-l+1);

int t = arr[l];

arr[l] = arr[rp];

arr[rp]=t;

return partition(arr,l,r);

}

int partition(intarr[],intl,int r)

{

intx,y,p,t;

x=l;

y=r;

p=arr[l];

while(x<y)

{

while(arr[x]<=p && x<=y)

x++;

while(arr[y]>p&& x<=y)

y--;

if(x<y)

{

t = arr[x];

arr[x] = arr[y];

arr[y]=t;

}

}

t = arr[l];

arr[l]=arr[y];

arr[y]=t;

return y;

}