Index

Experiment No: 1 **Date:** 04-02-2023

Experiment Name: Write a program to sort some data using Bubble sort.

Code:

```
// Bubble Sort
#include <stdio.h>
int main()
{
   int array[1000],size, swap;
   printf("Bubble Sort\n");
   printf("Enter number of elements: ");
   scanf("%d", &size);
   printf("\nEnter %d integers\n", size);
   for (int step = 0; step < size; step++)
      scanf("%d", &array[step]);
   for ( int step = 0 ; step < size - 1; step++)
   {
      for (int i = 0 ; i < size - step - 1; i++)
      {
         if (array[i] > array[i+1])
         {
            swap = array[i];
            array[i] = array[i+1];
            array[i+1] = swap;
         }
      }
   }
   printf("\nSorted list in ascending order:\n");
   for (int step = 0; step < size; step++)
      printf("%d\n", array[step]);
   return 0;
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\bubble sort.exe"
Bubble Sort
Enter number of elements: 5

Enter 5 integers
6
9
17
22
30

Sorted list in ascending order:
6
9
17
22
30

Process returned 0 (0x0)   execution time : 42.395 s
Press any key to continue.
```

Experiment No: 2 **Date:** 04-02-2023

Experiment Name: Write a program to sort some data using Insertion sort.

Code:

```
// Insertion Sort
#include <stdio.h>
int main()
{
   int size, i, temp;
   int array[1000];
   printf("Enter number of elements: ");
   scanf("%d", &size);
   printf("\nEnter %d integers\n", size);
   for (int step = 0; step < size; step++)
   {
      scanf("%d", &array[step]);
   }
   for (int step = 1 ; step <= size - 1; step++)
   {
      i = step;
      while ( i > 0 && array[i-1] > array[i])
      {
         temp = array[i];
         array[i] = array[i-1];
         array[i-1] = temp;
         i--;
      }
   }
   printf("\nSorted list in ascending order:\n");
   for (int step = 0; step <= size - 1; step++)
   {
      printf("%d\n", array[step]);
```

```
    }
    return 0;
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\insertion sort.exe"
Enter number of elements: 5

Enter 5 integers
6
9
17
22
30

Sorted list in ascending order:
6
9
17
22
30

Process returned 0 (0x0)   execution time : 10.911 s
Press any key to continue.
```

Experiment No: 3 **Date:** 11-02-2023

Experiment Name: Write a program to search specific data from a list using linear search.

Code:

```
// Linear Search
#include <stdio.h>
int main()
{
   int array[1000], search, i, num, count = 0;
   printf("Enter number of elements in array: ");
   scanf("%d", &num);
   printf("\nEnter %d numbers\n", num);
   for (i = 0; i < num; i++)
      scanf("%d", &array[i]);
   printf("\nEnter a number to search: ");
   scanf("%d", &search);
   for (i = 0; i < num; i++)
   {
      if (array[i] == search)
      {
         printf("%d is present at location %d.\n", search, i+1);
         count++;
      }
   }
   if (count == 0)
      printf("%d isn't present in the array.\n", search);
   else
      printf("\n%d is present %d times in the array.\n", search, count);
   return 0;
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\Linear search.exe"
Enter number of elements in array: 5

Enter 5 numbers
6
9
17
22
30

Enter a number to search: 17
17 is present at location 3.

17 is present 1 times in the array.

Process returned 0 (0x0)   execution time : 17.205 s
Press any key to continue.
```

Experiment No: 4 **Date:** 11-02-2023

Experiment Name: Write a program to search specific data from a list using binary search.

Code:

```
// Binary Search
#include <stdio.h>
int main()
{
   int c, first, last, middle, n, search, array[100];
   printf("Enter the number of elements: ");
   scanf("%d", &n);
   if (n<=0)
      printf("Invalid!");
   else
   {
      printf("\nEnter %d integers in ascending order\n", n);
      for (c = 0; c < n; c++)
         scanf("%d", &array[c]);
      printf("\nEnter value to find in the list: ");
      scanf("%d", &search);
      first = 0;
      last = n - 1;
      middle = (first+last)/2;
      while (first <= last)
      {
         if (array[middle] < search)
            first = middle + 1;
         else if (array[middle] == search)
         {
            printf("%d found at location %d\n", search, middle+1);
            break;
         }
```

```
        else
            last = middle - 1;
        middle = (first + last)/2;
    }
    if (first > last)
        printf("Not found!\n%d isn't present in the list\n", search);
  }
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\Binary search.exe"
Enter the number of elements: 5

Enter 5 integers in ascending order
6
9
17
22
30

Enter value to find in the list: 17
17 found at location 3

Process returned 0 (0x0)   execution time : 8.786 s
Press any key to continue.
```

Experiment No: 5 **Date:** 18-02-2023

Experiment Name: Write a program to perform some stack operation (push, pop, print) using array.

Code:

```
// Perform some stack operation (push, pop, print) using array
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main( )
{
   top=-1;
   printf("Enter the STACK size (maximum 100):");
   scanf("%d",&n);
   printf("\nEnter what operations you want to perform :");
   printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT\n");
   do
   {
      printf("\nEnter the Choice: ");
      scanf("%d",&choice);
      switch(choice)
      {
      case 1:
      {
         push( );
         break;
      }
      case 2:
      {
         pop( );
         break;
```

```
        }
        case 3:
        {
            display( );
            break;
        }
        case 4:
        {
            printf("\n\t EXIT POINT ");
            break;
        }
        default:
        {
            printf ("\n\tPlease Enter a Valid Choice between 1/2/3/4");
        }
        }
    }
    while(choice!=4);
    return 0;
}
void push( )
{
    if(top>=n-1)
    {
        printf("\n\tSTACK is FULL!");
    }
    else
    {
        printf("Enter a value you want to push in stack: ");
        scanf("%d",&x);
        top++;
```

```
        stack[top]=x;
    }
}
void pop( )
{
    if(top<=-1)
    {
        printf("\n\t Stack is EMPTY!");
    }
    else
    {
        printf("\nThe popped element is %d",stack[top]);
        top--;
    }
}
void display( )
{
    if(top>=0)
    {
        printf("\nThe elements in STACK are :\n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
    }
    else
    {
        printf("\nThe STACK is EMPTY!");
    }
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\Stack.exe"
Enter the STACK size (maximum 100):20

Enter what operations you want to perform :
         1. PUSH
         2. POP
         3. DISPLAY
         4. EXIT

Enter the Choice: 1
Enter a value you want to push in stack: 6

Enter the Choice: 1
Enter a value you want to push in stack: 9

Enter the Choice: 1
Enter a value you want to push in stack: 17

Enter the Choice: 3

The elements in STACK are :

17
9
6
Enter the Choice: 2

The popped element is 17
Enter the Choice: 3

The elements in STACK are :

9
6
Enter the Choice: 4

         EXIT POINT
Process returned 0 (0x0)   execution time : 19.603 s
Press any key to continue.
```

Experiment No: 6 **Date:** 25-02-2023

Experiment Name: Write a program to perform some queue operation (enqueue, dequeue, display) using array.

Code:

```
// Perform some queue operation (enqueue, dequeue, display) using array
#include <stdio.h>
#define SIZE 20
void enqueue();
void dequeue();
void display();
int items[SIZE], front = -1, rear = -1,choice;
int main( )
{
   printf("Choose the option you want to be performed :");
   printf("\n\t 1.ENQUEUE\n\t 2.DEQUEUE\n\t 3.DISPLAY\n\t 4.EXIT");
   printf("\n");
   do
   {
      printf("\nEnter the Choice:");
      scanf("%d",&choice);
      switch(choice)
      {
      case 1:
      {
         enqueue( );
         break;
      }
      case 2:
      {
         dequeue( );
         break;
```

```
        }
        case 3:
        {
            display( );
            break;
        }
        case 4:
        {
            printf("\n\t EXIT POINT ");
            break;
        }
        default:
        {
            printf ("\n\tPlease Enter a Valid Choice between 1/2/3/4");
        }
        }
    }
    while(choice!=4);
    return 0;
}
void enqueue()
{
    int item;
    if(rear == SIZE - 1)
        printf("Queue FULL! \n");
    else
    {
        if(front== - 1)
            front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &item);
```

```
        rear = rear + 1;
        items[rear] = item;
    }
}
void dequeue()
{
    if(front == - 1 || front > rear)
    {
        printf("Queue EMPTY! \n");
        return;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", items[front]);
        front = front + 1;
    }
}
void display()
{
    int i;
    if(front == - 1)
        printf("Queue is FULL! \n");
    else
    {
        printf("Queue is : \n");
        for(i = front; i <= rear; i++)
            printf("%d ", items[i]);
        printf("\n");
    }
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\Queue.exe"
Choose the option you want to be performed :
         1. ENQUEUE
         2. DEQUEUE
         3. DISPLAY
         4. EXIT

Enter the Choice:1
Inset the element in queue : 6

Enter the Choice:1
Inset the element in queue : 9

Enter the Choice:1
Inset the element in queue : 17

Enter the Choice:3
Queue is :
6 9 17

Enter the Choice:2
Element deleted from queue is : 6

Enter the Choice:2
Element deleted from queue is : 9

Enter the Choice:3
Queue is :
17

Enter the Choice:4

         EXIT POINT
Process returned 0 (0x0)   execution time : 27.770 s
Press any key to continue.
```

Experiment No: 7 **Date:** 04-03-2023

Experiment Name: Write a program to print Fibonacci series using recursion.

Code:

```
// Fibonaci Series
#include<stdio.h>
void Fibonacci(int num)
{
   static int num1=0,num2=1,num3;
   if(num>0)
   {
      num3 = num1 + num2;
      num1 = num2;
      num2 = num3;
      printf("%d ",num3);
      Fibonacci(num-1);
   }}
int main()
{
   int num;
   printf("Enter the number of elements to show: ");
   scanf("%d",&num);
   printf("\nFibonacci Series: ");
   printf("%d %d ",0,1);
   Fibonacci(num-2);
   printf("\n");
   return 0;
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\Fibonacci.exe"
Enter the number of elements to show: 9

Fibonacci Series: 0 1 1 2 3 5 8 13 21

Process returned 0 (0x0)   execution time : 9.785 s
Press any key to continue.
```

Experiment No: 8 **Date:** 04-03-2023

Experiment Name: Write a program to calculate power using recursion.

Code:

```
// Calculate power using recursion
#include <stdio.h>
double pow(double base, double exponent);
int main()
{
   double base, power;
   int exponent;
   printf("Enter base: ");
   scanf("%lf", &base);
   printf("Enter exponent: ");
   scanf("%d", &exponent);
   power = pow(base, exponent);
   printf("%.2lf ^ %d = %.2lf", base, exponent, power);
   return 0;
}
double pow(double base, double exponent)
{
   if(exponent == 0)
      return 1;
   else if(exponent > 0)
      return base * pow(base, exponent - 1);
   else
      return 1 / pow(base, -exponent);
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\power.exe"
Enter base: 2
Enter exponent: 7
2.00 ^ 7 = 128.00
Process returned 0 (0x0)   execution time : 8.086 s
Press any key to continue.
```

Experiment No: 9 **Date:** 18-03-2023

Experiment Name: Write a program to calculate permutation using recursion.

Code:

```
// Calculate permutation using recursion
#include <stdio.h>
#include <string.h>
int count=0;
void swap(char *x, char *y)
{
   char temp;
   temp = *x;
   *x = *y;
   *y = temp;
}
void permutation(char s[], int l, int r)
{
   if (l == r)
   {
      puts(s);
      count++;
   }
   else
   {
      for (int i = l; i <= r; i++)
      {
         swap(&s[l], &s[i]);
         permutation(s, l+1, r);
         swap(&s[l], &s[i]);
      }
   }
}
```

```
int main()
{
    char str[100];
    printf("Enter Expression: ");
    gets(str);
    int n = strlen(str);
    permutation(str, 0, n-1);
    printf("\nTotal Permutation:%d\n",count);
    return 0;
}
```

Output:

```
Select "C:\Users\USER\OneDrive\Desktop\DSA Code\Permutation.exe"
Enter Expression: Love
Love
Loev
Lvoe
Lveo
Levo
Leov
oLve
oLev
ovLe
oveL
oevL
oeLv
voLe
voeL
vLoe
vLeo
veLo
veoL
eovL
eoLv
evoL
evLo
eLvo
eLov

Total Permutation:24

Process returned 0 (0x0)   execution time : 9.664 s
Press any key to continue.
```

Experiment No: 10 **Date:** 18-03-2023

Experiment Name: Write a program to calculate combination using recursion.

Code:

```
// Calculate combination using recursion
#include <stdio.h>
int NCR (int n, int r)
{
   if (r == 0 || n == r)
   {
      return 1;
   }
   else
      return NCR (n - 1, r - 1) + NCR (n - 1, r);
}
int main ()
{
   int n,r;
   printf("Enter a number n: ");
   scanf("%d",&n);
   printf("Enter a number r: ");
   scanf("%d",&r);
   printf("\nValue of %dC%d = %d\n",n,r, NCR (n, r));
}
```

Output:

```
"C:\Users\USER\OneDrive\Desktop\DSA Code\Combination.exe"
Enter a number n: 17
Enter a number r: 9

Value of 17C9 = 24310

Process returned 0 (0x0)   execution time : 4.508 s
Press any key to continue.
```