PRACTICAL 3

1) Write a C Program to implement searching on ordered array of 10 integer value.

```
#include<stdio.h>
int main()
{
   int arr[10], x, count=0;
   printf("Enter any 10 numbers: \n");
   for(int i=0; i<10; i++)  {
      scanf("%d",&arr[i]);
   }
   printf("Enter number that you want to search: \n");
   scanf("%d", &x);
   for(int i=0; i<10; i++)
   {
      if(arr[i]==x)  {
         printf("-------------------------\n");
         printf("The number is at index: %d\n", i);
      }
   }
}
```

OUTPUT

Enter any 10 numbers:

1 2 3 4 5 6 7 8 9 10

Enter number that you want to search:

5

The number is at index: 4

PRACTICAL 3

2) Write a C Program to implement Stack with all necessary overflow and underflow condition (Use array as data structure). 1)PUSH 2) POP 3) DISPLAY.

```
#include<stdio.h>
int stack[10],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()  {
   top=-1;
   printf("Enter the size of STACK:\n");
   scanf("%d",&n);
   printf("STACK OPERATIONS USING ARRAY\n");
   printf("--------------------------------\n");
   printf("1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n");
   do   {
      printf("Enter your Choice from 1 to 4: \n");
      scanf("%d",&choice);
      switch(choice)  {
         case 1:   {
            push();
            break;
         }
         case 2: {
            pop();
            break;
         }
         case 3:  {
            display();
            break;
```

```
        }
        case 4:  {
           printf("\n\t EXIT POINT\n");
           break;
        }
        default:   {
           printf ("\n\t Please Enter a Valid Choice(1/2/3/4)\n");
        }
      }
   }
   while(choice!=4);
   return 0;
}
void push()  {
   if(top>=n-1)  {
      printf("STACK is over flow\n");
  }
   else  {
      printf("Enter a value to be pushed:");
      scanf("%d",&x);
      top++;
      stack[top]=x;
   }
}
void pop() {
   if(top<=-1)   {
      printf("Stack is under flow\n");
   }
   else   {
```

```
      printf("The popped elements is %d\n",stack[top]);
      top--;
   }
}
void display()  {
   if(top>=0)  {
      printf("The elements in STACK \n");
      for(i=top; i>=0; i--) {
         printf("|");
         printf("__%d__|\n",stack[i]);
      }
      printf("Press Next Choice\n");
   }
   else   {
      printf("The STACK is empty\n");
for(int i=0; i<n; i++) {
      printf("|____|\n");
   }
 }
}
```

OUTPUT

Enter the size of STACK:

5

STACK OPERATIONS USING ARRAY

1.PUSH

2.POP

3.DISPLAY

4.EXIT

Enter your Choice from 1 to 4:

1

Enter a value to be pushed:5

Enter your Choice from 1 to 4:

1

Enter a value to be pushed:4

Enter your Choice from 1 to 4:

1

Enter a value to be pushed:3

Enter your Choice from 1 to 4:

1

Enter a value to be pushed:2

Enter your Choice from 1 to 4:

1

Enter a value to be pushed:1

Enter your Choice from 1 to 4:

3

The elements in STACK

|__1__|

|__2__|

|__3__|

|__4__|

|__5__|

Press Next Choice

Enter your Choice from 1 to 4:

2

The popped elements is 1

Enter your Choice from 1 to 4:

2

The popped elements is 2

Enter your Choice from 1 to 4:

3

The elements in STACK

|__3__|

|__4__|

|__5__|

Press Next Choice

Enter your Choice from 1 to 4:

2

The popped elements is 3

Enter your Choice from 1 to 4:

2

The popped elements is 4

Enter your Choice from 1 to 4:

2

The popped elements is 5

Enter your Choice from 1 to 4:

2

Stack is under flow

Enter your Choice from 1 to 4:

3

The STACK is empty

|____|

|____|

|____|

|____|

|____|

Enter your Choice from 1 to 4:

4

EXIT POINT

3) Write a C Program to convert infix notation into its equivalent postfix notation using stack.

```
#include<stdio.h>
#include<ctype.h>
char stack[100];
int top = -1;
void push(char x)
{
   top++;
   stack[top] = x;
}
char pop()
{
   if(top == -1)
      return -1;
   else
      return stack[top--];
}
int priority(char x)
{
   if(x == '(')
      return 0;
   if(x == '+' || x == '-')
      return 1;
   if(x == '*' || x == '/')
      return 2;
   return 0;
}
int main()
```

```
{
   char exp[20];
   char *e, x;
   printf("Enter the expression : ");
   scanf("%s",exp);
   printf("----------------------------------------\n");
   printf("Expression in Infix form: %s\n", exp);
   printf("----------------------------------------\n");
   printf("Expression in Postfix form: ");
   e = exp;
   while(*e != '\0') {
      if(isalnum(*e)) { // it checks that is char is alphabet or not
        printf("%c ",*e);
      }
      else if(*e == '(') {
         push(*e);
      }
      else if(*e == ')') {
         while((x = pop()) != '('
           printf("%c ", x);
      }
      else {
         while(priority(stack[top]) >= priority(*e))
         printf("%c ",pop());
         push(*e);
      } e++;
   }
   while(top != -1)
   {
```

```
      printf("%c ",pop());
   }
}
```

OUTPUT

Enter the expression : A+B-C*D/E*F

Expression in Infix form: A+B-C*D/E*F

Expression in Postfix form: A B + C D * E / F * -

4) Write a c Program to implement Tower of Hanoi problem.

```
#include<stdio.h>
void towerofhanoi(int n, char src[6], char helper[6], char dest[6] )   {
   if(n==1) {
      printf("Transfer disc %d from %s to %s\n", n, src, dest);
      return;
   }
   towerofhanoi(n-1, src, dest, helper);
   printf("Transfer disc %d from %s to %s\n", n, src, dest);
   towerofhanoi(n-1, helper, src, dest);
}
int main()  {
   int n;
   do {
   printf("-----------------------------------------------\nEnter the number of disc: ");
   scanf("%d", &n);
   towerofhanoi(n, "S", "H", "D");
   } while(n!=0);
}
```

OUTPUT

Enter the number of disc: 1

Transfer disc 1 from S to D

Enter the number of disc: 2

Transfer disc 1 from S to H

Transfer disc 2 from S to D

Transfer disc 1 from H to D

Enter the number of disc: 3

Transfer disc 1 from S to D

Transfer disc 2 from S to H

Transfer disc 1 from D to H

Transfer disc 3 from S to D

Transfer disc 1 from H to S

Transfer disc 2 from H to D

Transfer disc 1 from S to D

Enter the number of disc: 0

5) Write a C Program to implement Infix to prefix conversion using stack.

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include <stdlib.h>
#define MAX 20
void push(int);
char pop();
void infix_to_prefix();
int precedence (char);
char stack[20],infix[20],prefix[20];
int top = -1;
int main()  {
   printf("\nINPUT THE INFIX EXPRESSION : ");
   scanf("%s",infix);
   infix_to_prefix();
   return 0;
}
void push(int pos) {
   if(top == MAX-1) {
   printf("\nOVERFLOW\n");
   }
   else {
   top++;
   stack[top] = infix[pos];
   }
}
char pop() {
   char ch;
```

```
    if(top < 0) {
        printf("\nSTACK UNDERFLOW\n");
    }
    else {
        ch = stack[top];
        top--;
        return(ch);
    }
    return 0;
}
void infix_to_prefix() {
    int i = 0,j = 0;
    strrev(infix);
        while(infix[i] != '\0') {
            if(infix[i] >= 'a' && infix[i] <= 'z') {
                prefix[j] = infix[i];
                j++;
                i++;
            }
            else if(infix[i] == ')') {
                push(i);
                i++;
            }
            else if(infix[i] == '(') {
                while(stack[top] != ')')    {
                        prefix[j] = pop();
                        j++;
                    }
                    pop();
```

```
            i++;
         }
         else {
            if(top == -1) {
               push(i);
               i++;
            }
            else if( precedence(infix[i]) < precedence(stack[top]))  {
               prefix[j] = pop();
               j++;
            while(precedence(stack[top]) > precedence(infix[i])) {
               prefix[j] = pop();
               j++;
               if(top < 0) {
                  break;
               }
            }
            push(i);
            i++;
         }
         else if(precedence(infix[i]) >= precedence(stack[top])) {
            push(i);
            i++;
         }
      }
   }
   while(top != -1) {
      prefix[j] = pop();
      j++;
```

```
    }
    strrev(prefix);
    printf("EQUIVALENT PREFIX NOTATION : %s ",prefix);
}
int precedence(char alpha) {
       if(alpha == '+' || alpha =='-') {
          return(1);
       }
       if(alpha == '*' || alpha =='/') {
          return(2);
       }
          return 0;
}
```

Output

INPUT THE INFIX EXPRESSION : a+b*c/d-e*f

EQUIVALENT PREFIX NOTATION : -+a/*bcd*ef

Question 1: Explain why stack is recursive data structure?

Thus, in recursion last function called needs to be completed first. Now Stack is a LIFO data structure i.e. (Last In First Out) and hence it is used to implement recursion. The High-level Programming languages, such as Pascal, C etc. that provides support for recursion use stack for book keeping.

Question 2: Real time applications of stack (at least 10 applications).

1. Converting infix to postfix expressions.
2. Undo/Redo button/operation in word processors.
3. Syntaxes in languages are parsed using stacks.
4. It is used in many virtual machines like JVM.
5. Forward-backward surfing in the browser.
6. History of visited websites.
7. Message logs and all messages you get are arranged in a stack.
8. Call logs, E-mails, Google photos' any gallery, YouTube downloads, Notifications (latest appears first).
9. Scratch cards earned after Google pay transaction.
10. Wearing/Removing Bangles, Pile of Dinner Plates, Stacked chairs.
11. Changing wearables on a cold evening, first in, comes out at last.
12. Last Hired, First Fired - which is typically utilized when a company reduces its workforce in an economic recession.
13. Loading bullets into the magazine of a gun. The last one to go in is fired first. Bam!
14. Java Virtual Machine.
15. Recursion.
16. Used in IDEs to check for proper parentheses matching.
17. Media playlist. To play previous and next song.

Question 3: What is complexity of push and pop for a stack implemented using array?

In push operation you add one element at the top of the stack so you make one step, so it takes constant time so push takes O(1).

In pop operation you remove one element from the top of the stack so you make one step, so it takes constant time so pop takes O(1).