Exp1: WAP to implement Stack menu driven program

```
#include<stdio.h>
#include<conio.h>
#define MAX 5
int st[5], top=-1;
void push(int st[], int val);
int pop(int st[]);
void display(int st[]);
int peek(int st[]);
int main()
{
   int val, option;
   clrscr();
   printf("Exp1\tSIA2\tRoll no.39\n");
   do
   {
      printf("-----------------------MAIN MENU-----------------------");
      printf("\n1.Push\t2.Pop\t3.Display\t4.Peek\t5.Exit");
      printf("\nEnter your choice: ");
      scanf("%d",&option);
      switch(option)
            {
         case 1: printf(" Enter the no to be pushed on the stack: ");
               scanf("%d",&val);
               push(st,val);
               break;

         case 2: val=pop(st);
               if( val!=-1)
                  printf(" The value to be deleted form the stack is: %d\n", val);
               break;
         case 3: display(st);
               break;
         case 4: val=peek(st);
               if(val!=-1)
                  printf(" The value stored at top of the stack is: %d\n",val);
               break;
         case 5: printf("Exiting!");
               break;
         default: printf("Please enter a correct choice!");
      }
   }while(option!=5);
   return 0;
}
void push(int st[], int val)
{
   if(top==MAX-1){
      printf(" STACK OVERFLOW\n");
   }else{
     top++;
     st[top]=val;
   }
}
int pop(int st[])
{
```

```
   int val;
   if(top==-1){
      printf(" STACK UNDERFLOW\n");
      return -1;
   }else{
       val=st[top];
       top--;
       return val;
   }
}
void display(int st[])
{
   int i;
   if(top==-1){
         printf(" STACK IS EMPTY\n");
   }else{
             for(i=0;i<=top;i++)
             {
                printf(" %d\n",st[i]);
      }
   }
}
int peek(int st[])
{
          if(top== -1){
                    printf(" STACK IS EMPTY\n");
                    return -1;
   }else{
             return st[top];
          }
}
```

```
Exp1    SIA2    Roll no.39
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 2
 STACK UNDERFLOW
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 1
 Enter the no to be pushed on the stack: 10
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 1
 Enter the no to be pushed on the stack: 20
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 1
 Enter the no to be pushed on the stack: 30
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 1
 Enter the no to be pushed on the stack: 40
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 1
 Enter the no to be pushed on the stack: 50
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 1
 Enter the no to be pushed on the stack: 60
 STACK OVERFLOW
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 4
 The value stored at top of the stack is: 50
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 2
 The value to be deleted form the stack is: 50
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice: 3
 10
 20
 30
 40
-----------------------MAIN MENU-----------------------
1.Push  2.Pop   3.Display       4.Peek  5.Exit
Enter your choice:
```

Exp2: WAP to implement Infix to Postfix transformation

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#define MAX 100
char st[MAX];
int top=-1;
void push(char st[], char);
char pop(char st[]);
void InfixtoPostfix(char source[], char target[]);
int getPriority(char);
int main()
{
   char infix[100], postfix[100];
   clrscr();
   printf("Exp2\tSIA2\tRoll no.39");
   printf("\n Enter any infix expression : ");
   gets(infix);
   strcpy(postfix, "");
   InfixtoPostfix(infix, postfix);
   printf("\n The corresponding postfix expression is : ");
   puts(postfix);
   getch();
   return 0;
}
void InfixtoPostfix(char source[], char target[])
{
   int i=0, j=0;
   char temp;
   strcpy(target, "");
   while(source[i]!='\0')
   {
      if(source[i]=='(')
      {
         push(st, source[i]);
         i++;
      }
      else if(source[i] == ')')
      {
         while((top!=-1) && (st[top]!='('))
         {
            target[j] = pop(st);
            j++;
         }
         if(top==-1)
         {
            printf("\n INCORRECT EXPRESSION");
            exit(1);
         }
         temp = pop(st);//remove left parenthesis
         i++;
      }
      else if(isdigit(source[i]) || isalpha(source[i]))
      {
```

```
            target[j] = source[i];
            j++;
            i++;
        }
        else if (source[i] == '+' || source[i] == '-' || source[i] == '*' ||
                 source[i] == '/' || source[i] == '%')
        {
            while( (top!=-1) && (st[top]!= '(') && (getPriority(st[top])> getPriority(source[i])))
            {
                target[j] = pop(st);
                j++;
            }
            push(st, source[i]);
            i++;
        }
        else
        {
            printf("\n INCORRECT ELEMENT IN EXPRESSION");
            exit(1);
        }
    }
    while((top!=-1) && (st[top]!='('))
    {
        target[j] = pop(st);
        j++;
    }
    target[j]='\0';
}
int getPriority(char op)
{
    if(op=='/' || op == '*' || op=='%')
        return 1;
    else if(op=='+' || op=='–')
        return 0;
}
void push(char st[], char val)
{
    if(top==MAX-1)
        printf("\n STACK OVERFLOW");
    else
    {
        top++;
        st[top]=val;
    }
}
char pop(char st[])
{
    char val=' ';
    if(top==-1)
        printf("\n STACK UNDERFLOW");
    else
    {
        val=st[top];
        top--;
    }
```

```
    return val;
}
```

```
Exp2    SIA2    Roll no.39
 Enter any infix expression : (A+B)-C*(D+E)

 The corresponding postfix expression is : AB+C-DE+*
```

Exp3: WAP to implement queue menu driven program

```
#include<stdio.h>
#include<conio.h>
const int max=3;
void enqueue(int);
int dequeue();
int peek();
void display();
int stack[2],rear=-1,front=-1;
void main()
{
    int val,n,m;
    clrscr();
    printf("Exp3\tSIA2\tRoll no.39");
    do
    {
        printf("\n-----------------------MAIN MENU-----------------------");
        printf("\n1.ENQUEUE\t2.DEQUEUE\t3.PEEK\t4.Display\t5.Exit\nEnter your choice: ");
        scanf("%d",&n);
        switch(n)
        {
        case 1:
            printf("Enter value to ENQUEUE: ");
            scanf("%d",&m);
            enqueue(m);
            break;
        case 2:
            val=dequeue();
            if(val!=0)
                printf("Value DEQUEUED is: %d ",val);
            break;
        case 3:
            printf("Peek value: ");
            peek();
            break;
        case 4:
            printf("DISPLAY\n");
            display();
            break;
        case 5:
            printf("EXIT");
            break;
        default:
            printf("INVALID");
        }
    }
    while(n!=5);
```

```
   getch();
}
void enqueue(int m)
{
   if(rear==max-1)
   {
      printf("Overflow");
   }
   else
   {
      if(front==-1)
         front=0;
      rear++;
      stack[rear]=m;
      printf("%d is ENQUEUED",m);
   }
}
int dequeue()
{
   int m=0;
   if(front==-1)
   {
      printf("Underflow");
   }
   else
   {
      m=stack[front];
      front++;
   }
   return m;
}
int peek()
{
   int m=0;
   if(rear==-1)
   {
      printf("Stack is empty");
   }
   else
   {
      m=stack[rear];
      printf("%d",m);
   }
   return m;

}
void display()
{
   int i;
   printf("Stack elements are: ");
   for(i=front; i<=rear; i++)
   {
      printf("%d\n",stack[i]);
   }
}
```

```
Exp3    SIA2    Roll no.39
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 2
Underflow
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 1
Enter value to ENQUEUE: 2
2 is ENQUEUED
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 1
Enter value to ENQUEUE: 3
3 is ENQUEUED
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 3
Peek value: 3
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 1
Enter value to ENQUEUE: 4
4 is ENQUEUED
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 1
Enter value to ENQUEUE: 5
Overflow
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 2
Value DEQUEUED is: 2
------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice: 4
DISPLAY
Stack elements are: 3
4

------------------------MAIN MENU-------------------------
1.ENQUEUE       2.DEQUEUE       3.PEEK  4.Display       5.Exit
Enter your choice:
```

Exp4: WAP to implement double ended queue menu driven program

```
#include<stdio.h>
#include<conio.h>
const int max=3;
void enqueue(int);
void frontenqueue(int);
int dequeue();
int reardequeue();
void display();
int queue[2],rear=-1,front=-1;
int main()
{
   int val,n,m;
   clrscr();
   printf("Exp4\tSIA2\tRoll no.39");
   do
   {
      printf("\n-----------------------MAIN MENU-----------------------");
      printf("\n1.ENQUEUE rear\t2.ENQUEUE front\t3.DEQUEUE rear\n4.DEQUEUE front\t5.Display\t6.EXIT\nEnter the
choice:");
      scanf("%d",&n);
      switch(n)
      {
      case 1:
         printf("Enter value to ENQUEUE from rear: ");
         scanf("%d",&m);
         enqueue(m);
         break;
      case 2:
         printf("Enter value to ENQUEUE from front: ");
         scanf("%d",&m);
         frontenqueue(m);
         break;
      case 3:
         if(rear==-1)
         {
            printf("UNDERFLOW");
         }
         else
         {
            val=reardequeue();
         }
         printf("Value DEQUEUED from rear is %d",val);
         break;
      case 4:
         if(front==-1)
         {
            printf("UNDERFLOW");
         }
         else
         {
            val=dequeue();
         }
         printf("Value DEQUEUED from front is %d",val);
         break;
```

```
        case 5:
            printf("DISPLAY\n");
            display();
            break;
        case 6:
            printf("EXIT");
            break;
        default:
            printf("INVALID");
        }
    }
    while(n!=6);
    getch();
}
void enqueue(int m)
{
    if(rear==(max-1))
    {
        printf("OVERFLOW");
    }
    else if(front==-1)
    {
        front=0;
        rear=0;
        queue[front]=m;
        printf("%d is ENQUEUED",m);
    }
    else
    {
        rear++;
        queue[rear]=m;
        printf("%d is ENQUEUED",m);
    }
}
int dequeue()
{
    int m=0;
    if(front==rear)
    {
        m=queue[front];
        front=-1;
        rear=-1;
        return m;
    }
    else
    {
        m=queue[front];
        front++;
    }
    return m;
}
void display()
{
    int i;
    printf("Queue elements are:\n");
```

```
    for(i=front; i<=rear; i++)
    {
        printf("%d\t",queue[i]);
    }
}
int reardequeue()
{
    int m=0;
    if(front==rear)
    {
        m=queue[front];
        front=-1;
        rear=-1;
    }
    else
    {
        m=queue[rear];
        rear--;
    }
    return m;
}
void frontenqueue(int m)
{
    if(front==0)
    {
        printf("OVERFLOW");
    }
    else if(rear==-1)
    {
        rear=0;
        front=0;
        queue[front]=m;
        printf("%d is ENQUEUED",m);
    }
    else
    {
        --front;
        queue[front]=m;
        printf("%d is ENQUEUED",m);
    }
}
```

```
Exp4    SIA2    Roll no.39
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:1
Enter value to ENQUEUE from rear: 33
33 is ENQUEUED
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:1
Enter value to ENQUEUE from rear: 55
55 is ENQUEUED
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:1
Enter value to ENQUEUE from rear: 77
77 is ENQUEUED
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:5
DISPLAY
Queue elements are:
33      55      77
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:4
Value DEQUEUED from front is 33
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:2
Enter value to ENQUEUE from front: 88
88 is ENQUEUED
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:5
DISPLAY
Queue elements are:
88      55      77
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:3
Value DEQUEUED from rear is 77
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:5
DISPLAY
Queue elements are:
88      55
------------------------MAIN MENU-----------------------
1.ENQUEUE rear  2.ENQUEUE front 3.DEQUEUE rear
4.DEQUEUE front 5.Display       6.EXIT
Enter the choice:
```

Exp5: WAP to implement linked list menu driven program

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
struct node
{
   int data;
   struct node *next;
};
struct node *start = NULL;
struct node *create_ll(struct node *start)
{
   struct node *new_node, *ptr;
   int num;
   printf("\n Enter -1 to end\n");
   printf(" Enter the data : ");
   scanf("%d", &num);
   while(num!=-1)
   {
      new_node = (struct node*)malloc(sizeof(struct node));
      new_node -> data=num;
      if(start==NULL)
      {
         new_node -> next = NULL;
         start = new_node;
      }
      else
      {
         ptr=start;
         while(ptr->next!=NULL)
            ptr=ptr->next;
         ptr->next = new_node;
         new_node->next=NULL;
      }
      printf(" Enter the data : ");
      scanf("%d", &num);
   }
   return start;
}
struct node *display(struct node *start)
{
   struct node *ptr;
   ptr = start;
   while(ptr != NULL)
   {
      printf("\t %d", ptr -> data);
      ptr = ptr -> next;
   }
   return start;
}
struct node *insert_beg(struct node *start)
{
   struct node *new_node;
   int num;
```

```
    printf("\n Enter the data : ");
    scanf("%d", &num);
    new_node = (struct node *)malloc(sizeof(struct node));
    new_node -> data = num;
    new_node -> next = start;
    start = new_node;
    return start;
}
struct node *insert_end(struct node *start)
{
    struct node *ptr, *new_node;
    int num;
    printf("\n Enter the data : ");
    scanf("%d", &num);
    new_node = (struct node *)malloc(sizeof(struct node));
    new_node -> data = num;
    new_node -> next = NULL;
    ptr = start;
    while(ptr -> next != NULL)
        ptr = ptr -> next;
    ptr -> next = new_node;
    return start;
}
struct node *insert_before(struct node *start)
{
    struct node *new_node, *ptr, *preptr;
    int num, val;
    printf("\n Enter the data : ");
    scanf("%d", &num);
    printf("\n Enter the value before which the data has to be inserted : ");
    scanf("%d", &val);
    new_node = (struct node *)malloc(sizeof(struct node));
    new_node -> data = num;
    ptr = start;
    while(ptr -> data != val)
    {
        preptr = ptr;
        ptr = ptr -> next;
    }
    preptr -> next = new_node;
    new_node -> next = ptr;
    return start;
}
struct node *insert_after(struct node *start)
{
    struct node *new_node, *ptr, *preptr;
    int num, val;
    printf("\n Enter the data : ");
    scanf("%d", &num);
    printf("\n Enter the value after which the data has to be inserted : ");
    scanf("%d", &val);
    new_node = (struct node *)malloc(sizeof(struct node));
    new_node -> data = num;
    ptr = start;
    preptr = ptr;
```

```
   while(preptr -> data != val)
   {
      preptr = ptr;
      ptr = ptr -> next;
   }
   preptr -> next=new_node;
   new_node -> next = ptr;
   return start;
}
struct node *delete_beg(struct node *start)
{
   struct node *ptr;
   ptr = start;
   start = start -> next;
   free(ptr);
   return start;
}
struct node *delete_end(struct node *start)
{
   struct node *ptr, *preptr;
   ptr = start;
   while(ptr -> next != NULL)
   {
      preptr = ptr;
      ptr = ptr -> next;
   }
   preptr -> next = NULL;
   free(ptr);
   return start;
}
struct node *delete_node(struct node *start)
{
   struct node *ptr, *preptr;
   int val;
   printf("\n Enter the value of the node which has to be deleted : ");
   scanf("%d", &val);
   ptr = start;
   if(ptr -> data == val)
   {
      start = delete_beg(start);
      return start;
   }
   else
   {
      while(ptr -> data != val)
      {
         preptr = ptr;
         ptr = ptr -> next;
      }
      preptr -> next = ptr -> next;
      free(ptr);
      return start;
   }
}
struct node *delete_after(struct node *start)
```

```
{
   struct node *ptr, *preptr;
   int val;
   printf("\n Enter the value after which the node has to deleted : ");
   scanf("%d", &val);
   ptr = start;
   preptr = ptr;
   while(preptr -> data != val)
   {
      preptr = ptr;
      ptr = ptr -> next;
   }
   preptr -> next=ptr -> next;
   free(ptr);
   return start;
}
struct node *delete_list(struct node *start)
{
   struct node *ptr;
   if(start!=NULL)
   {
      ptr=start;
      while(ptr != NULL)
      {
         printf("\n %d is to be deleted next", ptr -> data);
         start = delete_beg(ptr);
         ptr = start;
      }
   }
   return start;
}
struct node *sort_list(struct node *start)
{
   struct node *ptr1, *ptr2;
   int temp;
   ptr1 = start;
   while(ptr1 -> next != NULL)
   {
      ptr2 = ptr1 -> next;
      while(ptr2 != NULL)
      {
         if(ptr1 -> data > ptr2 -> data)
         {
            temp = ptr1 -> data;
            ptr1 -> data = ptr2 -> data;
            ptr2 -> data = temp;
         }
         ptr2 = ptr2 -> next;
      }
      ptr1 = ptr1 -> next;
   }
   return start;
}
int main()
{
```

```
int option;
clrscr();
printf("Exp5\tSIA2\tRoll no.39");
do
{
    printf("\n----------MAIN MENU----------");
    printf("\n 1: Create a list");
    printf("\n 2: Display the list");
    printf("\n 3: Add a node at the beginning");
    printf("\n 4: Add a node at the end");
    printf("\n 5: Add a node before a given node");
    printf("\n 6: Add a node after a given node");
    printf("\n 7: Delete a node from the beginning");
    printf("\n 8: Delete a node from the end");
    printf("\n 9: Delete a given node");
    printf("\n 10: Delete a node after a given node");
    printf("\n 11: Delete the entire list");
    printf("\n 12: Sort the list");
    printf("\n 13: EXIT");
    printf("\n Enter your option : ");
    scanf("%d", &option);
    switch(option)
    {
    case 1:
        start = create_ll(start);
        printf("\n LINKED LIST CREATED");
        break;
    case 2:
        start = display(start);
        break;
    case 3:
        start = insert_beg(start);
        break;
    case 4:
        start = insert_end(start);
        break;
    case 5:
        start = insert_before(start);
        break;
    case 6:
        start = insert_after(start);
        break;
    case 7:
        start = delete_beg(start);
        break;
    case 8:
        start = delete_end(start);
        break;
    case 9:
        start = delete_node(start);
        break;
    case 10:
        start = delete_after(start);
        break;
    case 11:
```

```
            start = delete_list(start);
            printf("\n LINKED LIST DELETED");
            break;
        case 12:
            start = sort_list(start);
            break;
        }
    }
    while(option !=13);
    getch();
    return 0;
}
```

Output1:

```
Exp5    SIA2    Roll no.39
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 1

 Enter -1 to end
 Enter the data : 20
 Enter the data : 30
 Enter the data : 10
 Enter the data : -1

 LINKED LIST CREATED
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 3

 Enter the data : 5

---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 4

 Enter the data : 1
```

Output2:

```
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 2
         5       20      30      10      1
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 5

 Enter the data : 25

 Enter the value before which the data has to be inserted : 10

---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 6

 Enter the data : 35

 Enter the value after which the data has to be inserted : 20
```

Output3:

```
--------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 2
         5       20      35      30      25      10      1
--------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 7

--------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 8
```

Output4:

```
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 2
         20      35      30      25      10
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 9

 Enter the value of the node which has to be deleted : 30

---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 2
         20      35      25      10
```

Output5:

```
--------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 10

 Enter the value after which the node has to deleted : 35

--------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 2
         20      35      10
--------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 12
```

Output6:

```
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 2
         10      20      35
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option : 11

 10 is to be deleted next
 20 is to be deleted next
 35 is to be deleted next
 LINKED LIST DELETED
---------------MAIN MENU---------------
 1: Create a list
 2: Display the list
 3: Add a node at the beginning
 4: Add a node at the end
 5: Add a node before a given node
 6: Add a node after a given node
 7: Delete a node from the beginning
 8: Delete a node from the end
 9: Delete a given node
 10: Delete a node after a given node
 11: Delete the entire list
 12: Sort the list
 13: EXIT
 Enter your option :
```

Exp6: WAP to implement different operations on linked list as copy, concatenate, split, reverse, count no. of nodes etc.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node*link;
};
struct node*start=NULL;
void create(struct node**);
void display(struct node*);
void split(struct node*,struct node**,struct node**);
int count(struct node*);
void rev(struct node**,struct node*);
void copy(struct node**,struct node*);
void concatenate(struct node**,struct node**);
void main()
{

   struct node*first=NULL;
   struct node*second=NULL;
   struct node*c=NULL;
   struct node*r=NULL;
   int t;
   clrscr();
   printf("Exp6\tSIA2\tRoll no.39\n");
   create (&start);
   printf("#Linkedlist is as follows\n");
   display(start);
   copy(&c,start);
   printf("\nThe copied linked list is as follows:\n");
   display(c);
   rev(&r,start);
   printf("\nLinkedlist is reversed as follows:\n");
   display(r);
   split(start,&first,&second);
   printf("\nLinkedlist is split into two\n");
   printf("The first linkedlist is as follows\n");
   display(first);
   printf("\nThe second linkedlist is as follows\n");
   display(second);
   concatenate(&first,&second);
   printf("\nThe split linkedlist is concatenated as follows:\n");
   display(first);
   t=count(c);
   printf("\nThe count is:%d",t);
   getch();
}
void display(struct node*q)
{
   struct node*temp;
   temp=q;
   while(temp!=NULL)
```

```
    {
        printf("%d->",temp->data);
        temp=temp->link;
    }
    printf("NULL");
}
void create(struct node**q)
{
    int n,num,i;
    struct node*temp;
    printf("Enter the no of elements to be inserted:");
    scanf("%d",&n);
    for(i=0; i<n; i++)
    {
        printf("Enter a no:");
        scanf("%d",&num);
        temp=(struct node*)malloc(sizeof(struct node));
        temp->data=num;
        temp->link=NULL;
        if(*q==NULL)
            *q=temp;
        else
        {
            struct node*p;
            p=*q;
            while(p->link!=NULL)
            {
                p=p->link;
            }
            p->link=temp;
        }
    }
}
void split(struct node*p,struct node**first,struct node**second)
{
    int numnodes,splitpt,i=0;
    struct node*temp,*newnode,*temp2;
    numnodes=count(p);
    temp=p;
    if(numnodes%2==0)
        splitpt=numnodes/2;
    else
        splitpt=numnodes/2+1;
    temp=p;
    while(i<splitpt)
    {
        newnode=(struct node*)malloc(sizeof(struct node));
        newnode->data=temp->data;
        if(i==0)
        {
            *first=newnode;
            temp2=newnode;
        }
        else
        {
```

```
            temp2->link=newnode;
            temp2=temp2->link;
        }
        temp=temp->link;
        i++;
    }
    temp2->link=NULL;
    while(temp!=NULL)
    {
        newnode=(struct node*)malloc(sizeof(struct node));
        newnode->data=temp->data;
        if(*second==NULL)
        {
            *second=newnode;
            temp2=newnode;
        }
        else
        {
            temp2->link=newnode;
            temp2=temp2->link;
        }
        temp=temp->link;
    }
    temp2->link=NULL;
}
int count(struct node*a)
{
    int count=0;
    while(a!=NULL)
    {
        count++;
        a=a->link;
    }
    return(count);
}
void rev(struct node**q,struct node*p)
{
    struct node*temp,*temp1;
    temp=p;
    while(temp!=NULL)
    {
        if(*q==NULL)
        {
            *q=(struct node*)malloc(sizeof(struct node));
            (*q)->link=NULL;
            temp1=*q;
        }
        else
        {
            *q=(struct node*)malloc(sizeof(struct node));
            (*q)->link=temp1;
            temp1=*q;
        }
        (*q)->data=temp->data;
        temp=temp->link;
```

```
   }
}
void copy(struct node**q,struct node*p)
{
   if ( p != NULL )
   {
      *q = malloc ( sizeof ( struct node ) ) ;

      ( *q ) -> data = p -> data ;
      ( *q ) -> link = NULL ;

      copy (  &( ( *q ) -> link ),p -> link) ;
   }
}
void concatenate(struct node**p,struct node**q)
{
   struct node*temp;
   if(*p==NULL)
      *p=*q;
   else
   {
      if(*q!=NULL)
      {
         temp=*p;
         while(temp->link!=NULL)
            temp=temp->link;
         temp->link=*q;
      }
   }
}
```

```
Exp6    SIA2    Roll no.39
Enter the no of elements to be inserted:3
Enter a no:5
Enter a no:8
Enter a no:10
#Linkedlist is as follows
5->8->10->NULL
The copied linked list is as follows:
5->8->10->NULL
Linkedlist is reversed as follows:
10->8->5->NULL
Linkedlist is split into two
The first linkedlist is as follows
5->8->NULL
The second linkedlist is as follows
10->NULL
The split linkedlist is concatenated as follows:
5->8->10->NULL
The count is:3
```

Exp7: WAP to construct expression tree using postfix expression

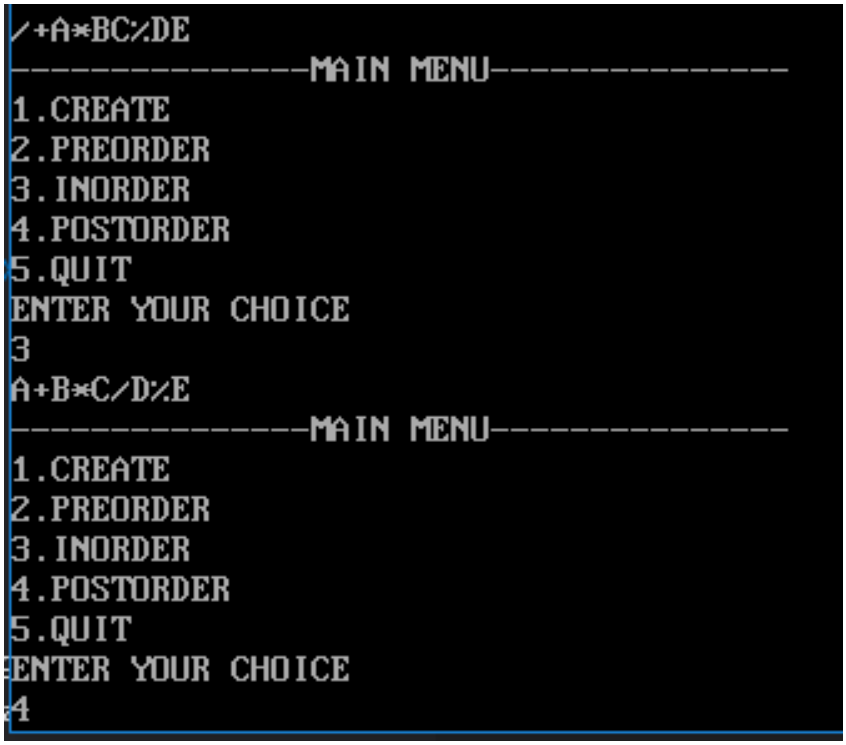
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
/*----------------------------------define tree node-------------------------------------*/
typedef struct treenode
{
   char data;
   struct treenode *left,*right;
} treenode;
/*----------------------------------define stack-------------------------------------*/
typedef struct stack
{
   treenode *data[20];
   int top;
} stack;
/*----------------------------------STACK OPERATIONS -------------------------------------*/
void init(stack *s)
{
   s->top=-1;
}
treenode *pop(stack *s)
{
   treenode *p;
   p=s->data[s->top];
   s->top=s->top-1;
   return(p);
}
void push(stack *s,treenode *p)
{
   s->top=s->top+1;
   s->data[s->top]=p;
}
int empty(stack *s)
{
   if(s->top==-1)
   {
      return(1);
   }
   return(0);
}
int full(stack *s)
{
   if(s->top==19)
      return(1);
   return(0);
}
/*----------------------------------TREE OPERATIONS----------------------------------*/
treenode *create();
void inorder(treenode *T);
void preorder(treenode *T);
void postorder(treenode *T);
/*----------------------------------MAIN FUNCTION----------------------------------*/
void main()
{
```

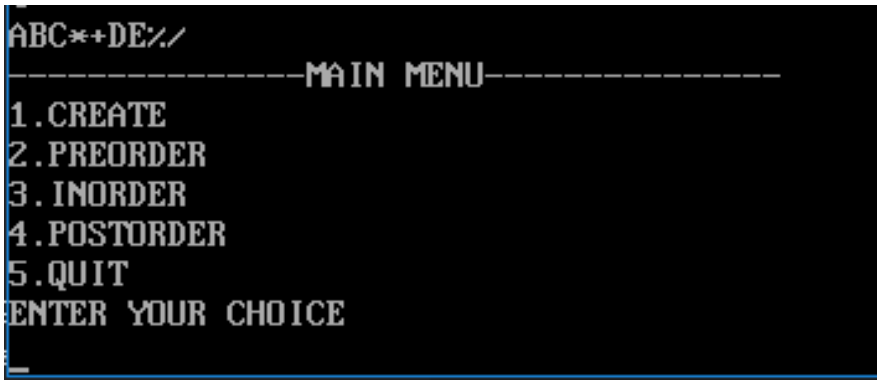
```
   treenode *root=NULL,*p;
   int x,op;
   printf("Exp7\tSIA2\tRoll no.39");
   clrscr();
   do
   {
      printf("\n----------MAIN MENU----------");
      printf("\n1.CREATE\n2.PREORDER\n3.INORDER\n4.POSTORDER\n5.QUIT\nENTER YOUR CHOICE\n");
      scanf("%d",&op);
      switch(op)
      {
      case 1:
         root=create();
         break;
      case 2:
         preorder(root);
         break;
      case 3:
         inorder(root);
         break;
      case 4:
         postorder(root);
         break;
      }
   }
   while(op!=5);
   getch();
}
/*----------TREE OPERATIONS DEFINATIONS----------*/
void inorder(treenode *T)
{
   if(T!=NULL)
   {
      inorder(T->left);
      printf("%c",T->data);
      inorder(T->right);
   }
}
void preorder(treenode *T)
{
   if(T!=NULL)
   {
      printf("%c",T->data);
      preorder(T->left);
      preorder(T->right);
   }
}
void postorder(treenode *T)
{
   if(T!=NULL)
   {
      postorder(T->left);
      postorder(T->right);
      printf("%c",T->data);
   }
```

```
}
treenode * create()
{
   char a[50];
   int i;
   treenode *p,*q,*root;
   stack s;
   init(&s);
   flushall();
   printf("ENTER A POSTFIX EXPRESSION\n");
   gets(a);
   for(i=0; a[i]!='\0'; i++)
   {
      if(isalnum(a[i]))                    //if operand then push to stack
      {
         p=(treenode *)malloc(sizeof(treenode));
         p->left=p->right=NULL;
         p->data=a[i];
         push(&s,p);
      }
      else                                 //if operator then pop two operands and perform operation
      {
         q=pop(&s);
         p=pop(&s);
         root=(treenode *)malloc(sizeof(treenode));
         root->left=p;
         root->right=q;
         root->data=a[i];
         push(&s,root);
      }
   }
   root=pop(&s);                 //remove the last element left in stack as root of tree
   return(root);
}
```

```
Exp7    SIA2    Roll no.39
---------------MAIN MENU---------------
1.CREATE
2.PREORDER
3.INORDER
4.POSTORDER
5.QUIT
ENTER YOUR CHOICE
1
ENTER A POSTFIX EXPRESSION
ABC**+DE%/

---------------MAIN MENU---------------
1.CREATE
2.PREORDER
3.INORDER
4.POSTORDER
5.QUIT
ENTER YOUR CHOICE
2
```





Exp8: WAP to implement graph menu driven program (DFS & BFS)

```
#include<stdio.h>
#include<conio.h>
int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();

void main()
{
   int n,i,s,ch,j;
   char c,dummy;
   clrscr();
   printf("Exp8\tSIA2\tRoll no.39");
   printf("\nENTER THE NUMBER VERTICES: ");
   scanf("%d",&n);
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=n; j++)
      {
         printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0: ",i,j);
         scanf("%d",&a[i][j]);
```

```
    }
  }
  printf("THE ADJACENCY MATRIX IS\n");
  for(i=1; i<=n; i++)
  {
    for(j=1; j<=n; j++)
    {
      printf(" %d",a[i][j]);
    }
    printf("\n");
  }
  do
  {
    for(i=1; i<=n; i++)
      vis[i]=0;
    printf("\nMENU");
    printf("\n1.B.F.S");
    printf("\n2.D.F.S");
    printf("\nENTER YOUR CHOICE: ");
    scanf("%d",&ch);
    printf("ENTER THE SOURCE VERTEX: ");
    scanf("%d",&s);
    switch(ch)
    {
    case 1:
      bfs(s,n);
      break;
    case 2:
      dfs(s,n);
      break;
    }
    printf("\nDO U WANT TO CONTINUE(Y/N)? ");
    scanf("%c",&dummy);
    scanf("%c",&c);
  }
  while((c=='y')||(c=='Y'));
}
//**************BFS(breadth-first search) code**************//
void bfs(int s,int n)
{
  int p,i;
  add(s);
  vis[s]=1;
  p=delete();
  if(p!=0)
    printf(" %d",p);
  while(p!=0)
  {
    for(i=1; i<=n; i++)
      if((a[p][i]!=0)&&(vis[i]==0))
      {
        add(i);
        vis[i]=1;
      }
    p=delete();
```

```
    if(p!=0)
        printf(" %d ",p);
  }
  for(i=1; i<=n; i++)
    if(vis[i]==0)
        bfs(i,n);
}

void add(int item)
{
  if(rear==19)
    printf("QUEUE FULL");
  else
  {
    if(rear==-1)
    {
        q[++rear]=item;
        front++;
    }
    else
        q[++rear]=item;
  }
}
int delete()
{
  int k;
  if((front>rear)||(front==-1))
    return(0);
  else
  {
    k=q[front++];
    return(k);
  }
}
//***************DFS(depth-first search) code******************//
void dfs(int s,int n)
{
  int i,k;
  push(s);
  vis[s]=1;
  k=pop();
  if(k!=0)
    printf(" %d ",k);
  while(k!=0)
  {
    for(i=1; i<=n; i++)
        if((a[k][i]!=0)&&(vis[i]==0))
        {
            push(i);
            vis[i]=1;
        }
    k=pop();
    if(k!=0)
        printf(" %d ",k);
  }
```

```
   for(i=1; i<=n; i++)
      if(vis[i]==0)
         dfs(i,n);
}
void push(int item)
{
   if(top==19)
      printf("Stack overflow ");
   else
      stack[++top]=item;
}
int pop()
{
   int k;
   if(top==-1)
      return(0);
   else
   {
      k=stack[top--];
      return(k);
   }
}
```

```
Experiment No.8          Batch: SIA2              Roll no.39
ENTER THE NUMBER VERTICES: 3
ENTER 1 IF 1 HAS A NODE WITH 1 ELSE 0: 1
ENTER 1 IF 1 HAS A NODE WITH 2 ELSE 0: 1
ENTER 1 IF 1 HAS A NODE WITH 3 ELSE 0: 0
ENTER 1 IF 2 HAS A NODE WITH 1 ELSE 0: 1
ENTER 1 IF 2 HAS A NODE WITH 2 ELSE 0: 0
ENTER 1 IF 2 HAS A NODE WITH 3 ELSE 0: 1
ENTER 1 IF 3 HAS A NODE WITH 1 ELSE 0: 0
ENTER 1 IF 3 HAS A NODE WITH 2 ELSE 0: 1
ENTER 1 IF 3 HAS A NODE WITH 3 ELSE 0: 1
THE ADJACENCY MATRIX IS
 1 1 0
 1 0 1
 0 1 1

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE: 1
ENTER THE SOURCE VERTEX: 2
 2 1  3
DO U WANT TO CONTINUE(Y/N)? y

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE: 2
ENTER THE SOURCE VERTEX: 2
 2  3  1
DO U WANT TO CONTINUE(Y/N)?
```

Exp9: WAP to implement quick sort and merge sort menu driven program

```
#include<stdio.h>
#include<conio.h>
#define size 10
int n=10;
int partition(int a[],int beg,int end);
void quick_sort(int a[],int beg,int end);
void merge(int a[],int,int,int);
void merge_sort(int a[],int,int);
void main()
{
   int arr[size],i,x;
   clrscr();
   printf("Exp9\tSIA2\tRoll no.39");
   do
   {
      printf("\n ENTER THE NO. OF ELEMENTS :\t");
      scanf("%d",&n);
      printf("ENTER THE ELEMENT :\t");
      for(i=0; i<n; i++)
      {
         scanf("%d",&arr[i]);
      }

      printf("\n ENTER YOUR CHOICE 1.QUICK SORT\t 2.MERGE SORT\t 3.EXIT\n");
      scanf("%d",&x);
      switch(x)
      {
      case 1:
         quick_sort(arr,0,n-1);
         printf("THE QUICK SORTED ARRAY IS:\n");
         for(i=0; i<n; i++)
            printf("%d \t",arr[i]);
         break;

      case 2:
         merge_sort(arr,0,n-1);
         printf("THE MERGE SORTED ARRAY IS:\n");
         for(i=0; i<n; i++)
         {
            printf("%d \t",arr[i]);
         }
         break;
      case 3:
         printf("EXIT");
         break;
      default:
         printf("INVALID");
      }
   }
   while(x!=3);
   getch();
}
int partition(int a[],int beg,int end)
{
```

```
    int left,right,temp,loc,flag;
    loc=left=beg;
    right=end;
    flag=0;
    while(flag!=1)
    {
        while((a[loc]<=a[right]) && (loc!=right))
            right--;
        if(loc==right)
            flag=1;
        else if(a[loc]>a[right])
        {
            temp=a[loc];
            a[loc]=a[right];
            a[right]=temp;
            loc=right;
        }
        if(flag!=1)
        {
            while((a[loc]>=a[left])&&(loc!=left))
                left++;
            if(loc==left)
                flag=1;
            else if(a[loc]<a[left])
            {
                temp=a[loc];
                a[loc]=a[left];
                a[left]=temp;
                loc=left;
            }
        }
    }
    return loc;
}
void quick_sort(int a[],int beg,int end)
{
    int loc,i;
    if(beg<end)
    {
        loc=partition(a,beg,end);
        quick_sort(a,beg,loc-1);
        quick_sort(a,loc+1,end);
    }
}
void merge_sort(int arr[],int beg,int end)
{
    int mid,i;
    int n=end;
    if(beg<end)
    {
        mid=(beg+end)/2;
        merge_sort(arr,beg,mid);
        merge_sort(arr,mid+1,end);
        merge(arr,beg,mid,end);
    }
```

```
}
void merge(int arr[],int beg,int mid,int end)
{
   int i=beg,j=mid+1,index=beg,temp[size],k;
   while((i<=mid) && (j<=end))
   {
      if(arr[i] < arr[j])
      {
         temp[index]=arr[i];
         i++;
      }
      else
      {
         temp[index]=arr[j];
         j++;
      }
      index++;
   }
   if(i<mid)
   {
      while(j<=end)
      {
         temp[index]=arr[j];
         j++;
         index++;
      }
   }
   else
   {
      while(i<=mid)
      {
         temp[index]=arr[i];
         i++;
         index++;
      }
   }
   for(k=beg; k<index; k++)
      arr[k]=temp[k];
}
```

```
Exp9    SIA2    Roll no.39
 ENTER THE NO. OF ELEMENTS :    6
ENTER THE ELEMENT :     9
3
4
6
1
8

 ENTER YOUR CHOICE 1.QUICK SORT  2.MERGE SORT    3.EXIT
1
THE QUICK SORTED ARRAY IS:
1       3       4       6       8       9
 ENTER THE NO. OF ELEMENTS :    6
ENTER THE ELEMENT :     3
9
5
6
1
7

 ENTER YOUR CHOICE 1.QUICK SORT  2.MERGE SORT    3.EXIT
2
THE MERGE SORTED ARRAY IS:
1       3       5       6       7       9
 ENTER THE NO. OF ELEMENTS :    1
ENTER THE ELEMENT :     1

 ENTER YOUR CHOICE 1.QUICK SORT  2.MERGE SORT    3.EXIT
3
```

Exp10: WAP to implement searching methods (Interpolation search)

```
#include <stdio.h>
#include <stdlib.h>
#include<stdio.h>
#include<conio.h>
# define size 10
void sort(int a[],int n);
int interpolation(int a[], int low, int high, int val);
void main()
{
        int a[size],i,n,pos,val;
```

```
	clrscr();
	printf("Exp10\tSIA2\tRoll no.39");
	printf("\nEnter the limit of array: ");
	scanf("%d",&n);
	printf("Enter the array elements: ");
	for(i=0;i<n;i++){
		scanf("%d",&a[i]);
	}
	sort(a,n);
	printf("\nSorted array is: ");
	for(i=0;i<n;i++){
		printf("%d\t",a[i]);
	}
	printf("\nEnter value to be searched: ");
	scanf("%d",&val);
	pos=interpolation(a,0,n-1,val);
	if(pos==-1)
	{
		printf("\nElement not found!");
	}else{
		printf("\nElement found at position: %d",pos+1);
	}
	getch();
}
void sort(int a[],int n)
{
	int i,j,temp;
	for(i=0;i<n;i++)
	{
		temp=a[i];
		j=i-1;
		while(temp<a[j] && j>=0)
		{
			a[j+1] = a[j];
			j--;
		}
		a[j+1] = temp;
	}
}
int interpolation(int a[], int low, int high, int val)
{
	int mid;
	while(low<=high)
	{
		mid = low+ (high - low)*((val - a[low])/(a[high] - a[low]));
		if(val == a[mid])
			return mid;
		if(val<a[mid])
			high = mid-1;
		else
			low=mid+1;
	}
	return -1;
}
```

Output Type-1

```
Exp10   SIA2    Roll no.39
Enter the limit of array: 9
Enter the array elements: 10
60
82
51
62
42
87
36
15

Sorted array is: 10     15      36      42      51      60      62      82      87
Enter value to be searched: 51

Element found at position: 5
```

Output Type-2

```
Exp10   SIA2    Roll no.39
Enter the limit of array: 9
Enter the array elements: 25
64
82
14
85
36
45
86
95

Sorted array is: 14     25      36      45      64      82      85      86      95
Enter value to be searched: 13

Element not found!
```

Exp11: WAP to implement hashing functions with different collision resolution technique.

```
#include<conio.h>
#include<stdio.h>
int a[10],h[10],i;
void hash1(int key);
void main()
{
   int n;
   clrscr();
   printf("Exp11\tSIA2\tRoll no.39");
   printf("\n Enter the number of elements:");
   scanf("%d",&n);
   printf("\n Enter the array elements:");
   for(i=0; i<n; i++)
   {
      scanf("%d",&a[i]);
   }
   for(i=0; i<10; i++)
   {
```

```
    h[i]=-1;
  }
  printf("Hash table before adding elements:");
  for(i=0; i<10; i++)
  {
    printf("%d\t",h[i]);
  }
  for(i=0; i<=9; i++)
  {
    hash1(a[i]);
  }
  printf("\nArray\tlocation\thashtable:\n");
  for(i=0; i<n; i++)
  {
    printf("%d\t%d\t\t%d\n",a[i],i,h[i]);
  }
  getch();
}
void hash1(int key)
{
  int index,i=0;
  do
  {
    index=(key+i)%10;
    if(h[index]!=-1)
      i++;
  }
  while(h[index]!=-1);
  h[index]=key;
}
```

```
Exp11   SIA2    Roll no.39
 Enter the number of elements:4

 Enter the array elements:10
40
20
50
Hash table before adding elements:-1    -1      -1      -1      -1      -1      -1      -1      -1      -1
Array   location        hashtable:
10      0               10
40      1               40
20      2               20
50      3               50
```