Ex. No:1A ARRAY IMPLEMENTATION OF STACK

Program:

```
#include<stdio.h>
#include<conio.h>
#define SIZE 10
void push(int);
void pop( );
void display();
void peek();
int stack[SIZE], top = -1;
void main()
{
 int value, choice;
 clrscr();
 while(1){
 printf("\n\n***** MENU *****\n");
 printf("1. Push\n2. Pop\n3. Display\n4. peek\n5. exit");
 printf("\nEnter your choice: ");
 scanf("%d",&choice);
 switch(choice){
        case 1: printf("Enter the value to be insert: ");
                scanf("%d",&value);
                push(value);
                break;
        case 2: pop();
                break;
        case 3: display();
                break;
        case 4: peek();
                break;
        case 5: exit(0);
default: printf("\nWrong selection!!! Try again!!!");
                }
          }
}
void push(int value)
{
 if(top == SIZE-1)
 printf("\nStack is Full!!! Insertion is not possible!!!");
 else{
        top++;
         stack[top] = value;
```

```
            printf("\nInsertion success!!!");
        }
}

void pop()
{
 if(top == -1)
 printf("\nStack is Empty!!! Deletion is not possible!!!");
 else{
         printf("\nDeleted : %d", stack[top]);
         top--;
       }
}
void display()
{
 if(top == -1)
 printf("\nStack is Empty!!!");
 else{
             int i;
             printf("\nStack elements are:\n");
            for(i=top; i>=0; i--)
            printf("%d\n",stack[i]);
        }
}
 void peek()
 {
 if(top==-1)
 printf("\nstack is empty!!!");
 else{
             printf("\ntop element of stack is %d",stack[top]);
        }
}
```

OUTPUT:

```
***** MENU *****
1. Push
2. Pop
3. Display
4. peek
5. exit

Enter your choice: 1
```

```
Enter the value to be insert: 45
Insertion Success!!!

***** MENU *****
1. Push
2. Pop
3. Display
4. peek
5. exit

Enter your choice: 1
Enter the value to be insert: 20
Insertion Success!!!

***** MENU *****
1. Push
2. Pop
3. Display
4. peek
5. exit

Enter your choice: 3
Stack elements are: 45  20

***** MENU *****
1. Push
2. Pop
3. Display
4. peek
5. exit

Enter your choice: 4
Top element of Stack is: 20

***** MENU *****
1. Push
2. Pop
3. Display
4. peek
5. exit

Enter your choice: 2
Deleted Element : 20
```

***** MENU *****
1. Push
2. Pop
3. Display
4. peek
5. exit

Enter your choice: 3
Stack elements are: 45

Ex.No: 1B ARRAY IMPLEMENTATION OF QUEUE

Program:

```
#include<stdio.h>
#include<conio.h>
#define SIZE 10
void enQueue(int);
void deQueue();
void display();
int queue[SIZE], front = -1, rear = -1;
void main()
{
  int value, choice;
  clrscr();
  while(1){
    printf("\n\n***** MENU *****\n");
    printf("1. Insertion\n2. Deletion\n3. Display\n4. Exit");
    printf("\nEnter your choice: ");
    scanf("%d",&choice);
    switch(choice){
        case 1: printf("Enter the value to be insert: ");
                scanf("%d",&value);
                enQueue(value);
                break;
        case 2: deQueue();
                break;
        case 3: display();
                break;
        case 4: exit(0);
        default: printf("\nWrong selection!!! Try again!!!");
                }
      }
}
void enQueue(int value)
{
  if(rear == SIZE-1)
  printf("\nQueue is Full!!! Insertion is not possible!!!");
  else{
       if(front == -1)
        front = 0;
        rear++;
       queue[rear] = value;
        printf("\nInsertion success!!!");
```

```
        }
}
void deQueue()
{
  if(front == -1||front>rear)
    printf("\nQueue is Empty!!! Deletion is not possible!!!");
  else{
          printf("\nDeleted : %d", queue[front]);
          front++;
       }
}
void display()
{
  if(front == -1||front>rear)
    printf("\nQueue is Empty!!!");
  else{
          int i;
          printf("\nQueue elements are:\n");
         for(i=front; i<=rear; i++)
            printf("%d\t",queue[i]);
         }
}
```

OUTPUT:

```
**Menu**
1.Insertion
2.deletion
3.display
4.exit
Enter the choice 1
Enter the value to be inserted 7
Insertion success

**Menu**
1.Insertion
2.deletion
3.display
4.exit
Enter the choice 1
Enter the value to be inserted 8
Insertion success
```

Menu
1.Insertion
2.deletion
3.display
4.exit
Enter the choice 2
deleted:7

Menu
1.Insertion
2.deletion
3.display
4.exit
Enter the choice 3
8

Ex.No:2A LINKED LIST IMPLEMENTATION OF STACK ADT

Program:

```
#include<stdio.h>
#include<conio.h>
struct node
{
int data;
struct node *link;
};
struct node *top=NULL;
void push();
void pop();
void display();
void exit();

void push()
{
struct node *temp;
temp=(struct node*)malloc(sizeof(struct node));
printf("\n Enter node data:");
scanf("%d",&temp->data);
temp->link=top;
top=temp;
}

void pop()
{
struct node  *temp;
if(top==NULL)
   printf("No element");
   else
   {
     temp=top;
     printf("The deleted element is%d",temp->data);
     top=top->link;
     temp->link=NULL;
     free(temp);
    }
}

void display()
{
```

```
  struct node  *temp;
  if(top==NULL)
  printf("\n Stack is empty");
  else
  {
    temp=top;
    while(temp!=NULL)
    {
          printf("%d\n",temp->data);
          temp=temp->link;
    }
  }
}
void main()
{
int choice;
clrscr();
do{
          printf("\n1.Push\n2.Pop\n3.Display\n4.Exit\n");
          printf("Enter your choice:");
          scanf("%d",&choice);
          switch(choice)
          {
           case 1:push();
                    break;
           case 2:pop();
                    break;
          case 3:display();
                    break;
          case 4:exit(0);
          default: printf("Invalid choice");
          `}
   }while(choice<6);
 getch();
 }
```

OUTPUT:

```
*****MENU******
1.Push
2.Pop
3.Display
4.Exit
```

```
Enter your choice 1
Enter node data 23

*****MENU******
1.Push
2.Pop
3.Display
4.Exit
Enter your choice 1
Enter node data 24

 *****MENU******
1.Push
2.Pop
3.Display
4.Exit
Enter your choice 2
The deleted element is 24

 *****MENU******
1.Push
2.Pop
3.Display
4.Exit
Enter your choice 1
Enter node data 25

 *****MENU******
1.Push
2.Pop
3.Display
4.Exit
Enter your choice 3
data is 23 25
```

Ex.No:2B LINKED LIST IMPLEMENTATION OF QUEUE ADT

Program:

```
#include<stdio.h>
#include<conio.h>
struct node
{
int data;
struct node * next;
}
*front=NULL,*rear=NULL;

void enqueue(int);
void dequeue();
void display();

void main()
{
int choice,value;
clrscr();
while(1)
{
printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit");
printf("\n enter your choice");
scanf("%d",&choice);
switch(choice)
{
case 1:
        printf("Enter the value to be inserted:");
        scanf("%d",&value);
        enqueue(value);
        break;
case 2:
        dequeue();
        break;
case 3:
        display();
        break;
case 4:
        exit(0);
default:
printf("Wrong selection!!!");
}
```

```
}
}
void enqueue(int value)
{
struct node * newnode;
newnode=(struct node *)malloc(sizeof(struct node));
newnode->data=value;
newnode->next=NULL;
if(front==NULL)
front=rear=newnode;
else
{
  rear->next=newnode;
  rear=newnode;
}
  printf("\nInsertion success");
}
void dequeue()
{
  if(front==NULL)
      printf("\nQueue is empty");
  else
     {
          struct node*temp=front;
          front=front->next;
          printf("\nDeleted element:%d",temp->data);
          free(temp);
     }
}
void display()
{
  if(front==NULL)
      printf("Empty");
  else
  {
    struct node*temp=front;
 while(temp->next!=NULL)
{
printf("%d",temp->data);
temp=temp->next;
}
printf("%d",temp->data);
} }
```

OUTPUT:

*****MENU******
1.Insert
2.Delete
3.Display
4.Exit
enter your choice 1
enter the value to be inserted 23

*****MENU******
1.Insert
2.Delete
3.Display
4.Exit
enter your choice 1
enter the value to be inserted 24

*****MENU******
1.Insert
2.Delete
3.Display
4.Exit
enter your choice 2
deleted element is:23

*****MENU******
1.Insert
2.Delete
3.Display
4.Exit
enter your choice 1
enter the value to be inserted 25

*****MENU******
1.Insert
2.Delete
3.Display
4.Exit
enter your choice 3
24 25

Ex.No: 3A ARRAY IMPLEMENTATION OF LIST ADT

Program:

```
#include<stdio.h>
#include<conio.h>
#define maxsize 10
int list[maxsize],n;
void Create();
void Insert();
void Delete();
void Display();
void Search();
void main()
{
int choice;
clrscr();
do
{
printf("\n Array Implementation of List\n");
printf("\t1.create\n");
printf("\t2.Insert\n");
printf("\t3.Delete\n");
printf("\t4.Display\n");
printf("\t5.Search\n");
printf("\t6.Exit\n");
printf("\nEnter your choice:\t");
scanf("%d",&choice);
switch(choice)
{
case 1:  Create();
         break;
case 2:  Insert();
         break;
case 3:  Delete();
         break;
case 4:   Display();
         break;
case 5:   Search();
         break;
case 6:   exit(1);
default:  printf("\nEnter option between 1 -  6\n");
                                  break;
}
```

```
}while(choice<7);
}
void Create()
{
int i;
printf("\nEnter the number of elements to be added in the list:\t");
scanf("%d",&n);
printf("\nEnter the array elements:\t");
for(i=0;i<n;i++)
                    scanf("%d",&list[i]);
Display();
}
void Insert()
{
int i,data,pos;
printf("\nEnter the data to be inserted:\t");
scanf("%d",&data);
printf("\nEnter the position at which element to be inserted:\t");
scanf("%d",&pos);
for(i = n-1 ; i >= pos-1 ; i--)
                      list[i+1] = list[i];
list[pos-1] = data;
n+=1;
Display();
}
void Delete( )
{
int i,pos;
printf("\nEnter the position of the data to be deleted:\t");
scanf("%d",&pos);
printf("\nThe data deleted is:\t %d", list[pos-1]);
for(i=pos-1;i<n-1;i++)
 list[i]=list[i+1];
n=n-1;
Display();
}
void Display()
{
int i;
printf("\n**********Elements in the array**********\n");
for(i=0;i<n;i++)
printf("%d\t",list[i]);
}
```

```
void Search()
{
int search,i,count = 0;
printf("\nEnter the element to be searched:\t");
scanf("%d",&search);
for(i=0;i<n;i++)
{
if(search == list[i])
  {
    count++;
  }
}
if(count==0)
printf("\nElement not present in the list");
else
printf("\nElement present in the list");
}
```

OUTPUT:

Array implementation of list

*****MENU******
1.create
2.Insert
3.Delete
4.Display
5.Search
6.exit
enter your choice:1
enter the number of elements to be added:3
enter the array elements: 21 22 23
 Elements in the array
 21 22 23

 Array implementation of list

*****MENU******
1.create
2.Insert
3.Delete
4.Display
5.Search

6.exit
enter your choice:2
enter the data to be inserted:24
enter the position at which element to be inserted:2
Elements in the array
21 24 22 23

Array implementation of list

*****MENU******
1.create
2.Insert
3.Delete
4.Display
5.Search
6.exit
enter your choice:3
enter the position of the data to be deleted:2
the data deleted is:24
Elements in the array
21 22 23

Array implementation of list

*****MENU******
1.create
2.Insert
3.Delete
4.Display
5.Search
6.exit
enter your choice:4
Elements in the array
21 22 23

Array implementation of list

*****MENU******
1.create
2.Insert
3.Delete
4.Display
5.Search
6.exit

enter your choice:5
Enter the element to be searched:21
Element present in the list

Array implementation of list

*****MENU******
1.create
2.Insert
3.Delete
4.Display
5.Search
6.exit
enter your choice:5
Enter the element to be searched:24
Element not present in the list

Ex.No:3B LINKED LIST IMPLEMENTATION OF LIST ADT

Program:

```
#include<stdio.h>
#include<conio.h>
struct node
{
 int data;
 struct node* link;
};
 struct node *root=NULL;

void addatend();
void display();
void addatbegin();
void delete();

void main()
{
  int ch,len;
  while(1)
  {
    printf("\n 1. addatend");
    printf("\n 2. display");
    printf("\n 3. addatbegin");
    printf("\n 4. To find the length");
    printf("\n 5. delete");
    printf("\n 6. quit");
    printf("\n Enter your choice:");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1: addatend();
              break;
      case 2: display();
              break;
      case 3: addatbegin();
              break;
      case 4: len=length();
              printf("\n The length of the linked list is %d",len);
              break;
      case 5: delete();
              break;
```

```
      case 6: exit(0);
      default: printf("\n Invalid");

    }
  }
}
void addatend()
{
  struct node* temp;
  temp=(struct node*)malloc(sizeof(struct node));
  printf("\n Enter the node data:");
  scanf("%d",&temp->data);
  temp->link=NULL;
  if(root==NULL)
  {
    root=temp;
  }
  else
  {
   struct node* p;
   p=root;
   while(p->link!=NULL)
   {
     p=p->link;
   }
   p->link=temp;
  }
}
void display()
  {
    struct node* temp;
    temp=root;
    if(temp==NULL)
    {
       printf("List is empty");
    }
    else
    {
     while(temp!=NULL)
     {
       printf("%d-->",temp->data);
       temp=temp->link;
     }
```

```
     printf("\n\n");
    }
}

void addatbegin()
{
    struct node* temp;
    temp=(struct node*)malloc(sizeof(struct node));
    printf("\n Enter the node data");
    scanf("%d",&temp->data);
    temp->link=NULL;
    if(root==NULL)
    {
    root=temp;
    }
    else
    {
    temp->link=root;
    root=temp;
    }
}
int length()
{
  int count=0;
  struct node* temp;
  temp=root;
  while(temp!=NULL)
  {
    count++;
    temp=temp->link;
  }
  return count;
}
void delete()
{
 struct node *temp;
 int loc;
 printf("\n Enter loc to delete");
 scanf("%d",&loc);
 if(loc>length())
 {
   printf("\n Invalid location");
 }
```

```
 else if(loc==1)
 {
 temp=root;
 root=temp->link;
 temp->link=NULL;
 free(temp);
 }
 else
 {
   struct node* p=root,*q;
   int i=1;
   while(i<loc-1)
   {
     p=p->link;
     i++;
   }
   q=p->link;
   p->link=q->link;
   q->link=NULL;
   free(q);
 }
}
```

OUTPUT:

```
1.addatend
2.display
3.addatbegin
4.to find the length
5.delete
6.quit
enter your choice:1
enter the node data:23


1.addatend
2.display
3.addatbegin
4.to find the length
5.delete
```

6.quit
enter your choice:1
enter the node data:56

1.addatend
2.display
3.addatbegin
4.to find the length
5.delete
6.quit
enter your choice:2
23-->56-->

1.addatend
2.display
3.addatbegin
4.to find the length
5.delete
6.quit
enter your choice:3
enter the node data:12

1.addatend
2.display
3.addatbegin
4.to find the length
5.delete
6.quit
enter your choice:2
12-->23-->56-->

1.addatend
2.display
3.addatbegin
4.to find the length
5.delete

6.quit
enter your choice:4
the length of the linked list is 3

1.addatend
2.display
3.addatbegin
4.to find the length
5.delete
6.quit
enter your choice:5
enter loc to delete:1

1.addatend
2.display
3.addatbegin
4.to find the length
5.delete
6.quit
enter your choice:2
23-->56-->

Ex.No:4A STACK APPLICATION: ARITHMETIC EXPRESSION EVALUATION

Program:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define MAX 20
int top = -1;
char stack[MAX];
char pop();
void push(char item);
int prcd(char symbol)
{
switch(symbol)
{
case '+':
case '-':
        return 2;
        break;
case '*':
case '/':
        return 4;
        break;
case '^':
case '$':
        return 6;
        break;
case '(':
case ')':
case '#':
        return 1;
        break;
}
}
int isoperator(char symbol)
{
switch(symbol)
{
case '+':
case '-':
case '*':
case '/':
case '^':
```

```
case '$':
case '(':
case ')':
return 1;
break;
default:
return 0;
}
}
void convertip(char infix[],char postfix[])
{
int i,symbol,j = 0;
stack[++top] = '#';
for(i=0;i<strlen(infix);i++)
{
symbol = infix[i];
if(isoperator(symbol) == 0)
{
postfix[j] = symbol;
j++;
}
else
{
if(symbol == '(')
push(symbol);
else if(symbol == ')')
{
while(stack[top] != '(')
{
postfix[j] = pop();
j++;
}
pop();
}
else
{
if(prcd(symbol) > prcd(stack[top]))
push(symbol);
else
{
while(prcd(symbol) <= prcd(stack[top]))
{
postfix[j] = pop();
```

```
j++;
}
push(symbol);
} }
} }
while(stack[top] != '#')
{
postfix[j] = pop();
j++;
}
postfix[j] = '\0';
}
int main()
{
char infix[20],postfix[20];
printf("Enter the valid infix string: ");
scanf("%s",infix);
convertip(infix, postfix);
printf("The corresponding postfix string is: ");
puts(postfix);
getch();
}
void push(char item)
{
top++;
stack[top] = item;
}
char pop()
{
char a;
a = stack[top];
top--;
return a;
}
```

OUTPUT:

Enter the valid infix string : a+b*c-d/e
The corresponding postfix string is: abc*+de/-

Ex. No: 4 B APPLICATIONS OF LINKED LIST: POLYNOMIAL MANIPULATION

Program:

```
#include<stdio.h>
#include<malloc.h>
#include<conio.h>
struct link
{
int coeff;
int pow;
struct link *next;
};
struct link *poly1=NULL,*poly2=NULL,*poly=NULL;
void create(struct link *node)
{
char ch;
do
{
printf("\nEnter coeff:");
scanf("%d",&node->coeff);
printf("\nEnter power:");
scanf("%d",&node->pow);
node->next=(struct link*)malloc(sizeof(struct link));
node=node->next;
node->next=NULL;
printf("\ncontinue(y/n):");
ch=getch();
}
while(ch=='y'||ch=='Y');
}
void show(struct link *node)
{
while(node->next!=NULL)
{
printf("%dx^%d",node->coeff,node->pow);
node=node->next;
if(node->next!=NULL)
printf("+");
}
}
void polyadd(struct link *poly1,struct link *poly2,struct link *poly)
{
while(poly1->next&&poly2->next)
```

```
{
if(poly1->pow>poly2->pow)
{
poly->pow=poly1->pow;
poly->coeff=poly1->coeff;
poly1=poly1->next;
}
else if(poly1->pow<poly2->pow)
{
poly->pow=poly2->pow;
poly->coeff=poly2->coeff;
poly2=poly->next;
}
else
{
poly->pow=poly1->pow;
poly->coeff=poly1->coeff+poly2->coeff;
poly1=poly1->next;
poly2=poly2->next;
}
poly->next=(struct link *)malloc(sizeof(struct link));
poly=poly->next;
poly->next=NULL;
}
while(poly1->next||poly2->next)
{
if(poly1->next)
{
poly->pow=poly1->pow;
poly->coeff=poly1->coeff;
poly1=poly1->next;
}
if(poly2->next)
{
poly->pow=poly2->pow;
poly->coeff=poly2->coeff;
poly2=poly2->next;
}
poly->next=(struct link *)malloc(sizeof(struct link));
poly=poly->next;
poly->next=NULL;
}
}
```

```
main()
{
char ch;
clrscr();
do{
poly1=(struct link *)malloc(sizeof(struct link));
poly2=(struct link *)malloc(sizeof(struct link));
poly=(struct link *)malloc(sizeof(struct link));
printf("\nEnter the 1st number:");
create(poly1);
printf("\nEnter the 2nd number:");
create(poly2);
printf("\n1st number:");
show(poly1);
printf("\n2nd number:");
show(poly2);
polyadd(poly1,poly2,poly);
printf("\nAdded Polynomial:");
show(poly);
printf("\nAdd two more numbers:");
ch=getch();
}
while(ch=='y'||ch=='Y');
}
```

OUTPUT:

```
Enter the 1st number:
Enter coeff: 5
Enter power: 3
Continue(y/N): N
Enter the 2nd number:
Enter coeff: 4
Enter power: 2
Continue(y/N): N
1st number: 5x^3
2nd number: 4x^2
Added polynomial: 5x^3+4x^2
```

Ex.No:5 IMPLEMENTATION OF BINARY TREES AND OPERATIONS

Program:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
int data;
struct node *left;
struct node *right;
}node;
int count=1;
node *insert(node *tree,int digit)
{
if(tree == NULL)
{
tree = (node *)malloc(sizeof(node));
tree->left = tree->right=NULL;
tree->data = digit;
count++;
}
else if(count%2 == 0)
tree->left = insert(tree->left, digit);
else
tree->right = insert(tree->right, digit);
return tree;
}
void preorder(node *t)
{
if(t != NULL)
{
printf(" %d", t->data);
preorder(t->left);
preorder(t->right);
}
}
void postorder(node *t)
{
if(t != NULL)
{
postorder(t->left);
postorder(t->right);
printf(" %d", t->data);
```

```
}  }
void inorder(node *t)
{
if(t != NULL)
{
inorder(t->left);
printf(" %d", t->data);
inorder(t->right);
}  }
main()
{
node *root = NULL;
int digit;
puts("Enter integer:To quit enter 0");
scanf("%d", &digit);
while(digit != 0)
{
root=insert(root,digit);
scanf("%d",&digit);
}
printf("\nThe preorder traversal of tree is:\n");
preorder(root);
printf("\nThe inorder traversal of tree is:\n");
inorder(root);
printf("\nThe postorder traversal of tree is:\n");
postorder(root);
getch();
return 0;
}
```

OUTPUT

```
Enter integer:
To quit enter 0
12 4 6 9 14 17 3 19 0
The preorder traversal of tree is: 12 4 9 17 19 6 14 3
The inorder traversal of tree is: 19 17 9 4 12 6 14 3
The postorder traversal of tree is: 19 17 9 4 3 14 6 12
```