1.Calendar pgm

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Day
{
 char* dayName;
 int date;
 char* activity;
};
struct Day* createCalendar()
{
 struct Day* calendar = (struct Day*)malloc(7 * sizeof(struct Day));
 int i;
for (i = 0; i < 7; i++)
{
 calendar[i].dayName = (char*)malloc(20 * sizeof(char));
 calendar[i].activity = (char*)malloc(100 * sizeof(char));
 }
 return calendar;
}
void readCalendarData(struct Day* calendar)
{
 int i;
 for(i = 0; i < 7; i++)
{
 printf("Enter the day name for Day %d: ", i + 1);
 scanf("%s",calendar[i].dayName);
printf("Enter the date for Day %d: ", i + 1);
scanf("%d",&calendar[i].date);
printf("Enter the activity for Day %d: ", i + 1);
```

```
scanf("%s",calendar[i].activity);
 }
}
void displayCalendar(struct Day* calendar)
{
printf("Weekly Activity Report:\n\n");
 int i;
 for ( i = 0; i < 7; i++)
{
 printf("Day %d: %s\n", i + 1, calendar[i].dayName);
 printf("Date: %d\n",
calendar[i].date);
 printf("Activity: %s\n", calendar[i].activity);
 printf("\n");
 }
}
int main()
{
 struct Day* calendar = createCalendar();
 readCalendarData(calendar);
 displayCalendar(calendar);
 int i;
 for (i = 0; i < 7; i++)
{
 free(calendar[i].dayName);
 free(calendar[i].activity);
 }
 free(calendar);
 return 0;
 }
```

2.String pgm

```
#include<stdio.h>
void read();
void match();
char STR[100],PAT[100],REP[100],ANS[100];
int c,i,j,k,m,flag=0;
main()
{
read();
match();
}
void read()
{
printf("enter the main string STR:");
gets(STR);
printf("enter pattern string PAT:");
gets(PAT);
printf("enter replace string REP:");
gets(REP);
}
void match()
{
c=i=j=k=m=0;
while(STR[c]!='\0')
{
if(STR[m]==PAT[i])
{
i++;m++;
flag=1;
if(PAT[i]=='\0')
{
```

```
for(k=0;REP[k]!='\0';k++,j++)
ANS[j]=REP[k];
i=0;
c=m;
}
}
else
{
ANS[j]=STR[c];
j++;c++;
m=c;
i=0;
}
}
if(flag==0)
printf("pattern not found");
else
{
ANS[j]='\0';
printf("resultant string is %s",ANS);
}
}
```

3.Push pop pgm

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 4
intstack[MAX],top=-1,item;
void push();
void pop();
void palindrome();
void display();
void main()
{
int choice;
while(1)
{
printf("------- STACK OPERATIONS ------\n");
printf("1.push\n 2.pop\n 3.palindrome\n 4.display\n 5.exit\n");
printf("enter choice");
scanf("%d",&choice);
switch(choice)
{
case 1:push();
break;
case 2:pop();
break;
case3:palindrome();
break;
case 4:display();
break;
case 5:exit(0);
break;
default:printf("invalidchoice\n");
```

```
break;
}}}
void push()
{
if(top==MAX-1)
printf("stackoverflow");
else
{
printf("enter the item to be pushed\n");
scanf("%d",&item);
top=top+1;
stack[top]=item;
}}
void pop()
{
if(top==-1)
printf("stackunderflow");
else
{
item=stack[top];
top=top-1;
printf("deleted item is %d",item);
}}
void display()
{
int i;
if(top==-1)
printf("stack is empty");
else
{
for(i=top;i>=0;i--)
```

```
printf("%d\t",stack[i]);
}}
void palindrome()
{
int num[10],i=0,k,flag=1;
k=top;
while(k!=-1)
num[i++]=stack[k--];
for(i=0;i<=top;i++)
{
if(num[i]==stack[i])
continue;
else
flag=0;
}
if(top==-1)
printf("stack is empty");
else
{
if(flag)
printf("palindrome");
else
printf("not a palindrome");
}
}
```

4.Infix to postfix

```
#include<stdio.h>
#include<ctype.h>
#define SIZE 50
chars[SIZE];
int top=-1;
void push(char elem)
{
s[++top]=elem;
}
char pop()
 {
return s[top--];
}
int pr(char elem)
 {
switch(elem)
{
case '#':return 0;
case '(':return 1;
case '+':
case '-':return 2;
case '*':
case '/':
case '%':return 3;
case '^':return 4;
 }}
void main()
{
charinfix[50],postfix[50],ch,elem;
int i=0,k=0;
```

```
printf("enter the infix expression\n");
gets(infix);
push('#');
while((ch=infix[i++])!='\0')
{
if(ch=='(')
push(ch);
else if(isalnum(ch))
postfix[k++]=ch;
else if(ch==')')
{
while(s[top]!='(')
postfix[k++]=pop();
elem=pop();
}
else
{
while(pr(s[top])>=pr(ch))
postfix[k++]=pop();
push(ch);
}}
while(s[top]!='#')
postfix[k++]=pop();
postfix[k]='\0';
printf("infix expression is %s\n postfix expression is %s\n",infix,postfix);
}
```

5(a).suffix expression

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define MAX 50
char post[MAX];
intstack[MAX],top=-1,i;
void pushstack(int);
voidcalculator(char);
main()
{
printf("entersuffix expression\n");
gets(post);
for(i=0; i<strlen(post); i++)
{
if(post[i]>'0'&&post[i]<='9')
pushstack(i);
else
calculator(post[i]);
}
printf("result=%d\n",stack[top]);
}
void pushstack(int i)
{
top=top+1;
stack[top]=(int)(post[i]-48);
}
void calculator(char c)
{
int a,b,ans;
```

```
b=stack[top--];
a=stack[top--];
switch(c)
{
case '+':ans=a+b;break;
case '-':ans=a-b;break;
case '*':ans=a*b;break;
case '/':ans=a/b;break;
case'%':ans=a%b;break;
case '^':ans=pow(a,b);break;
default :printf("wrong input\n");
exit(0);
}
top++;
stack[top]=ans;
}
```

5(b).tower of henoi

```
#include<stdio.h>
void tower(int n,char frompeg,char topeg,char auxpeg); int
n;
void main()
{
printf("Enter the no. of discs: \n");
scanf("%d",&n);
printf("the number of moves in tower of henoi problem\n");
tower(n,'A','C','B');
}
void tower(int n,char frompeg,char topeg,char auxpeg)
{
if(n==1)
{
printf("move disk1 from %C to %C\n ",frompeg,topeg);
return;
}
tower(n-1,frompeg,auxpeg,topeg);
printf("move disk%d from %C to %C\n",n,frompeg,topeg);
tower(n-1,auxpeg,topeg,frompeg);
}
```

6.queue pgm

```
#include<stdio.h>
#include<stdlib.h>
#define MAX5
char q[MAX],item;
int f=0,r=-1,count=0;
void insert();
void delete();
void display();
main()
{
int ch;
while(1)
{
printf("1.insert 2.delete 3.display 4.exit \n");
printf("enter choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:getchar();insert();
break;
case 2:delete();
break;
case 3:display();
break;
case 4:exit(0);
default :printf("Invalid choice\n");
break;
}}}
void insert()
{
```

```
if(count==MAX)
printf("queue overflow\n");
else
{
printf("enterthe itemto be inserted\n");
scanf("%c",&item);
r=(r+1)%MAX;
q[r]=item;
count++;
}}
void delete()
{
if(count==0)
printf("queue underflow\n");
else
{
printf("deleted itemis%c\n",q[f]);
f=(f+1)%MAX;
count--;
}
}
void display()
{
int j=f,i;
if(count==0)
printf("queue is empty\n");
else
{
printf("contents of circularqueue\n");
for(i=1;i<=count;i++)
{
```

```
printf("%c\t",q[j]);
j=(j+1)%MAX;
}
printf("total number of items=%d\n",count);
}}
```