7. Generate 3 address code

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

int i,j,ch,l,addr=100;
char exp[10],exp1[10],exp2[10],id1[5],id2[5],op[5];

void strrev(char str1[], int index, int size)
{
   char temp;

   temp = str1[index];
   str1[index] = str1[size - index];
   str1[size - index] = temp;

   if (index == size / 2)
   {
      return;
   }
   strrev(str1, index + 1, size);
}

void pm()
{
        strrev(exp,0,l-1);
        j=l-i-1;
        strncat(exp1,exp,j);
        strrev(exp1,0,strlen(exp1)-1);
        printf("\ntemp = %s\ntemp1 = %c%ctemp\n",exp1,exp[j+1],exp[j]);
}

void divii()
{
        strncat(exp1,exp,i+2);
        printf("\ntemp = %s\ntemp1 = temp%c%c\n",exp1,exp[i+2],exp[i+3]);
}

void plus()
{
        strncat(exp1,exp,i+2);
        printf("\ntemp = %s\ntemp1 = temp%c%c\n",exp1,exp[i+2],exp[i+3]);
}

void main()
{

        while(1)
        {
                printf("\nEnter 1. assignment\n2. arithmetic\n3. relational\n4. Exit\n");
                scanf("%d",&ch);
                switch (ch)
                {
                        case 1:
                                printf("Enter the expression: ");
                                scanf("%s",exp);
                                l=strlen(exp);
                                exp2[0]='\0';
                                i=0;
                                while(exp[i]!='=')
                                {
                                        i++;
                                }
                                strncat(exp2,exp,i);
                                strrev(exp,0,strlen(exp)-1);
                                exp1[0]='\0';
                                strncat(exp1,exp,l-(i+1));
                                strrev(exp1,0,strlen(exp1)-1);
                                printf("\ntemp = %s\n%s = temp\n",exp1,exp2);
                                break;
                        case 2:
                                printf("Enter the expression: ");
                                scanf("%s",exp);
                                l=strlen(exp);
                                exp1[0]='\0';
                                for(i=0;i<l;i++)
                                {
                                        if(exp[i]=='+'||exp[i]=='-')
                                        {
                                                if(exp[i+2]=='/'||exp[i+2]=='*')
                                                {
                                                        pm();
                                                        break;
                                                }
                                                else
                                                {
                                                        plus();
                                                        break;
                                                }
                                        }
                                        else if(exp[i]=='/'||exp[i]=='*')
                                        {
                                                divii();
                                                break;
                                        }
                                }
                                break;
                        case 3:
                                printf("Enter the expression: ");
                                scanf("%s%s%s",&id1,&op,&id2);

        if(((strcmp(op,"<")==0)||(strcmp(op,">")==0)||(strcmp(op,"<=")==0)||(strcmp(op,">=")==0)||(strcmp(op,"=
=")==0)||(strcmp(op,"!=")==0))==0)
                                        printf("Expression error");
                                else
                                {
                                        printf("\n%d\tif%s%s%s goto %d",addr,id1,op,id2,addr+3);
                                        addr++;
                                        printf("\n%d\t T:=0",addr);
                                        addr++;
                                        printf("\n%d\t goto %d",addr,addr+2);
                                        addr++;
                                printf("\n%d\t T:=1",addr);
                            }
                            break;
                        case 4:
                            exit(0);
                    }
        }
}
```

8a. Code optimization dead code and common expression elimination

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
 struct op
{
   char l;
   char r[20];
}
op[10], pr[10];

void main()
{
  int a, i, k, j, n, z = 0, m, q;
  char * p, * l;
  char temp, t;
  char * tem;
  clrscr();
  printf("enter no of values");
  scanf("%d", & n);
  for (i = 0; i < n; i++)
  {
            printf("\tleft\t");
            op[i].l = getche();
            printf("\tright:\t");
            scanf("%s", op[i].r);
  }
  printf("intermediate Code\n");
  for (i = 0; i < n; i++)
  {
             printf("%c=", op[i].l);
            printf("%s\n", op[i].r);
  }
  for (i = 0; i < n - 1; i++)
  {
              temp = op[i].l;
            for (j = 0; j < n; j++)
            {
                        p = strchr(op[j].r, temp);
                        if (p)
                        {
                                    pr[z].l = op[i].l;
                                    strcpy(pr[z].r, op[i].r);
                                    z++;

                         }
            }
  }
  pr[z].l = op[n - 1].l;
  strcpy(pr[z].r, op[n - 1].r);
  z++;
  printf("\nafter dead code elimination\n");
  for (k = 0; k < z; k++)
 {
              printf("%c\t=", pr[k].l);
             printf("%s\n", pr[k].r);
 }

  //sub expression elimination
  for (m = 0; m < z; m++)
{
    tem = pr[m].r;
    for (j = m + 1; j < z; j++)
    {
            p = strstr(tem, pr[j].r);
            if (p)
           {
                   t = pr[j].l;
                   pr[j].l = pr[m].l;
                   for (i = 0; i < z; i++)
                    {
                            l = strchr(pr[i].r, t);
                           if (l)
                            {
                                   a = l - pr[i].r;
                                   pr[i].r[a] = pr[m].l;
                           }
                   }
           }
    }
  }
  printf("eliminate common expression\n");
  for (i = 0; i < z; i++) {
    printf("%c\t=", pr[i].l);
    printf("%s\n", pr[i].r);
  }
  // duplicate production elimination

  for (i = 0; i < z; i++)
 {
           for (j = i + 1; j < z; j++)
           {
                   q = strcmp(pr[i].r, pr[j].r);
                   if ((pr[i].l == pr[j].l) && !q)

                   {
                           pr[i].l = '\0';
                           strcpy(pr[i].r, '\0');
                   }
           }
  }
  printf("optimized code");
  for (i = 0; i < z; i++)
{
           if (pr[i].l != '\0')
           {
                    printf("%c=", pr[i].l);
                   printf("%s\n", pr[i].r);
           }
}
 getch();
}
```

8b. Constant folding

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void main()
{
        char s[20];
        char flag[20]="//Constant";
        char result,equal,operator;
        double op1,op2,interrslt;
        int a,flag2=0;
        FILE *fp1,*fp2;

        fp1 = fopen("input.txt","r");
        fp2 = fopen("output.txt","w");

        fscanf(fp1,"%s",s);
        while(!feof(fp1))
        {
                if(strcmp(s,flag)==0)
                {
                        flag2 = 1;
                }
                if(flag2==1)
                {
                        fscanf(fp1,"%s",s);
                        result=s[0];
                        equal=s[1];
                        if(isdigit(s[2]) && isdigit(s[4]))
                        {
                                if(s[3]=='+'||'-'||'*'||'/')
                                {
                                        operator = s[3];
                                        switch(operator)
                                        {
                                                case '+':
                                                        interrslt = (s[2]-48)+(s[4]-48);
                                                        break;
                                                case '-':
                                                        interrslt = (s[2]-48)-(s[4]-48);
                                                        break;
                                                case '*':
                                                        interrslt = (s[2]-48)*(s[4]-48);
                                                        break;
                                                case '/':
                                                        interrslt = (s[2]-48)/(s[4]-48);
                                                        break;
                                                default:
                                                        interrslt = 0;
                                                        break;

                                        }
                                        fprintf(fp2,"/*Constant Folding */\n");
                                        fprintf(fp2,"%c = %lf\n",result,interrslt);
                                        flag2 = 0;
```

```
                    }
                }
                else
                {
                    fprintf(fp2,"Not Optimized\n");
                    fprintf(fp2,"%s\n",s);
                }
            }
            else
            {
                fprintf(fp2,"%s\n",s);
            }
            fscanf(fp1,"%s",s);
        }
        fclose(fp1);
        fclose(fp2);

}
```

9.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

typedef struct
{
        char var[10];
        int alive;
}
regist;

regist preg[10];

void substring(char exp[],int st, int end)
{
        int i,j=0;
        char dup[10]="";
        for(i=st;i<end;i++)
        dup[j++]=exp[i];

        dup[j]='\0';

        strcpy(exp,dup);

}

int getreg(char var[])
{
        int i;
        for(i=0;i<10;i++)
        {
                if(preg[i].alive==0)
                {
                        strcpy(preg[i].var,var);
                        break;
                }
        }
        return (i);
}

void getvar(char exp[], char v[])
{
        int i,j=0;
        char var[10]="";
        for(i=0;exp[i]!='\0';i++)
                if(isalpha(exp[i]))
                        var[j++]=exp[i];
                else
                        break;
        strcpy(v,var);
}

void main()
{
        char basic[10][10],var[10][10],fstr[10],op;
        int i,j,k,reg,vc = 0,flag=0;
```

```
printf("Enter 3 address code:\n");
for(i=0;;i++)
{
        gets(basic[i]);
        if(strcmp(basic[i],"exit")==0)
                break;
}
printf("\nAssembly Code: \n");
for(j=0;j<i;j++)
{
        getvar(basic[j],var[vc++]);
        strcpy(fstr,var[vc-1]);
        substring(basic[j],strlen(var[vc-1])+1,strlen(basic[j]));
        getvar(basic[j],var[vc++]);
        reg = getreg(var[vc-1]);
        if(preg[reg].alive==0)
        {
                printf("\nMOV R%d,%s",reg,var[vc-1]);
                preg[reg].alive = 1;
        }
        op = basic[j][strlen(var[vc-1])];
        substring(basic[j],strlen(var[vc-1])+1,strlen(basic[j]));
        getvar(basic[j],var[vc++]);
        switch(op)
        {
                case '+':
                        printf("\nAdd ");
                        break;
                case '-':
                        printf("\nSub ");
                        break;
                case '*':
                        printf("\nMul ");
                        break;
                case '/':
                        printf("\nDiv ");
                        break;
        }
        flag = 1;
        for(k=0;k<=reg;k++)
        {
                if(strcmp(preg[k].var,var[vc-1])==0)
                {
                        printf("R%d, R%d",k,reg);
                        preg[k].alive=0;
                        flag=0;
                        break;
                }
        }
        if(flag)
        {
                printf("%s,R%d",var[vc-1],reg);
                printf("\nMov %s,R%d",fstr,reg);
        }

        strcpy(preg[reg].var,var[vc-3]);
}
```

```
}
```