9.
```
#include <stdio.h> #include <stdlib.h>
#include <string.h>  int S[100];  int A[10][100];
int len, nf, i, j, v, f, h, k, ctn;
void read()
{ int i;  printf("Enter The No Of Frames : ");
scanf("%d", &nf);  printf("Enter The Length : ");
scanf("%d", &len);  printf("Enter The String : ");
for (i = 0; i < len; i++)    scanf("%d", &S[i]);  }
void display()
{ for (i = 0; i < len; i++)  printf("%d ", S[i]);
printf("\n\n");     for (i = 0; i <= nf; i++)
{ for (j = 0; j < len; j++)
{ if (A[i][j] == -1)
printf(" ");     else
printf("%d ", A[i][j]); }
if (i == nf - 1)
printf("\n");   printf("\n");  }
printf("No of Hits : %d\n", h);
printf("No of Faults : %d\n", (len - h));  }
void setInit()
{  for (i = 0; i < nf; i++)
A[i][0] = -1;  A[0][0] = S[0];  }
void alg(int type)
{ h = 0;      for (j = 1; j < len; j++)
```

```
{  f = 0;   int C[10] = { 0 };
for (i = 0; i < nf; i++)
{  A[i][j] = A[i][j - 1];
if (A[i][j] == S[j])
f = 1; }                if (f)
{ A[nf][j] = 1;  h++;  }
else       {  if (type == 1) // LRU
{  ctn = 0;
for (i = j - 1; i >= 0 && ctn < nf - 1; i--)
for (k = 0; k < nf; k++)
if (A[k][j] == S[i])
{ C[k] = 1;  ctn++;  }   v = 0;
for (i = 0; i < nf; i++)
if (C[i] == 0)
{  v = i;  break;  }   A[v][j] = S[j];
} else // FIFO
{  A[v][j] = S[j];
v = (v + 1) % nf; }  }  }  }
int main()              {  int ch;              while (1)
{printf("1)FIFO 2)LRU 3)EXIT\nEnter The Choice :
");   scanf("%d", &ch);   v = 1;
switch (ch)
{ case 1:  read();   setInit();  alg(0);  display(); break;
case 2:  read();  setInit();  alg(1);  display();  break;
case 3:
exit(0);   }   }  }
```

4.
```
#include<stdio.h>  #include<string.h>
#include<stdlib.h>
CharG[6][10]={"E+T","T*F","(E)","T","F","
id"};
char L[6]={'E','T','F','E','T','F'};
char stk[20]="$";  char inpt[20];
char pr[20],temp[20];
int i,j,k,f,len,lens,lenr,top=1,index1;
void extract()
{ for(k=0;k<lenr;k++)
temp[k]=stk[lens-lenr+k+1]; temp[k]='\0';
}  void print()  {
printf("%s\t\t%s$\t\t%s\n",stk,inpt,pr);
}  void shift()  { if(inpt[len-1]==' ')
{  sprintf(pr,"ERROR");  print();
exit(0);  }
stk[top++]=inpt[i];
if(inpt[i]=='i' && inpt[i+1]=='d')
{  stk[top++]=inpt[i+1];  inpt[i+1]=' ';
inpt[i]=' ';  sprintf(pr,"SHIFT id");  i++;  }
else
{  sprintf(pr,"SHIFT %c",inpt[i]);
inpt[i]=' ';  }  print();  }
void reduce()
{  int l=strlen(stk);
stk[l-lenr]=L[index1]; top=l-lenr+1;
```

```
for(k=top;k<20;k++)
stk[k]='\0';  sprintf(pr,"REDUCE %c->%s",L[index1],G[index1]);  print();  }
int main1()
{  printf("The given GRAMMAR is \nE->E+T|T\nT->T*F|F\nF->(E)|id\n");
printf("Enter The String : ");
scanf("%[^\n]",inpt);  len=strlen(inpt);
printf("STACK\t\tINPUT\t\t\tACTION\n");
print(pr);  i=0;  lens=strlen(stk)-1;
while(i<len || lens>0)
{  f=0;  lens=strlen(stk)-1;
if(stk[1]=='E' && lens==1 && inpt[len-1]==' ')    {  sprintf(pr,"ACCEPT");
print();  break;  }
if(lens==0)
{  shift();  i++;  continue;  }
for(j=0;j<6;j++)
{  lenr=strlen(G[j]);  if(lens<lenr)
continue;  extract();
if(strcmp(temp,G[j])==0)
{  if((j==0 || j==3) && inpt[i]=='*')
break;  f=1;  index1=j;  break;  }  }
if(f==1)
{  reduce();  }       else
{  shift();  i++;  continue;  }  } }
```

7. `#include<stdio.h>  #include<stdlib.h>`

```
int bt[10], rbt[10], at[10] = { 0 },
ct[10], wt[10], tat[10];
int choice, tq, n;  void roundRobin();
void srtf();  void readBT() {
printf("Enter Burst Time : ");
for (int i = 0; i < n; i++) {
scanf("%d", &bt[i]);  rbt[i] = bt[i]; } }
void readAT() {
printf("Enter Arrival Time : ");
for (int i = 0; i < n; i++)
scanf("%d", &at[i]);  }
void display() {
int swt = 0, stat = 0;
for (int i = 0; i < n; i++) {
tat[i] = ct[i] - at[i];
wt[i] = tat[i] - bt[i];
swt += wt[i];  stat += tat[i];  }
printf("PNO\tAT\tBT\tCT\tTAT\tWT\n");
for (int i = 0; i < n; i++)
printf("%d\t%d\t%d\t%d\t%d\t%d\n", i, at[i],
bt[i], ct[i], tat[i], wt[i]);
printf("Average TAT : %f\n", (float) stat /
n);  printf("Average WT : %f\n", (float) swt /
n);  }
int main() {  setbuf(stdout,NULL);
for (;;) {
printf("1)RR\n2)SRTF\n3)EXIT\n");
printf("Enter Choice : ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("ROUND ROBIN\n"); printf("Enter The
Number Of Processes : ");
scanf("%d", &n);  readBT();
printf("Enter Time Quantum : ");
```

```
scanf("%d", &tq);
roundRobin();  break;
case 2:
printf("SRTF\n");  printf("Enter The
Number Of Processes : ");  scanf("%d",
&n);  readBT(); readAT();  srtf();
break;
case 3:
exit(0);  }  }
return 0;  }
void roundRobin() {
int count = 0, i, time = 0;
while (1) {
for (i = 0; i < n; i++) {
if (rbt[i] > tq) {
rbt[i] -= tq;  time += tq;
} else if (rbt[i] != 0) {
time += rbt[i];
count++;  rbt[i] = 0;
ct[i] = time;  }  }
if (count == n)
break;  }  display();  }
void srtf() {  int count = 0, i, time;
rbt[9] = 999;
for (time = 0; count != n; time++) {
int smallest = 9;
for (i = 0; i < n; i++) {
if (at[i] <= time && rbt[i] < rbt[smallest] &&
rbt[i] > 0)
smallest = i;  }
rbt[smallest]--;  if (rbt[smallest] == 0) {
count++;  ct[smallest] = time + 1;  }  }
display();  }
```

8. #include <stdio.h> #include <stdlib.h>

```
struct process
{ int alloc[5], max[5], need[5], finished;
} p[10];  int avail[5], req[5], work[5], sseq[10];
int np, nr;          void input()
{ int i, j, chk = 0;
printf("Enter The No Of Processes : ");
scanf("%d", &np);
printf("Enter The No Of Resources : ");
scanf("%d", &nr);
printf("Enter Availability Matrix : \n");
for (i = 0; i < nr; i++)  scanf("%d", &avail[i]);
printf("Enter Allocated Matrix : \n");
for (i = 0; i < np; i++)  for (j = 0; j < nr; j++)
scanf("%d", &p[i].alloc[j]);
printf("Enter Max Matrix : \n");
for (i = 0; i < np; i++)  for (j = 0; j < nr; j++)
{ scanf("%d", &p[i].max[j]);
p[i].need[j] = p[i].max[j] - p[i].alloc[j];
if (p[i].need[j] < 0)          chk = 1;  }
if (chk)
printf("Allocation must be Less than Max\n");  }
int safe() {   int flag, sp = 0, i, j;
for (i = 0; i < nr; i++)       work[i] = avail[i];
for (i = 0; i < np; i++)       p[i].finished = 0;
while (sp != np) {     flag = 0;
for (i = 0; i < np; i++)
{ if (p[i].finished)          continue;   int less = 1;
for (j = 0; j < nr; j++)
if (p[i].need[j] > work[j])             less = 0;
if (less)       {  p[i].finished = 1;  flag = 1;
sseq[sp++] = i;          for (j = 0; j < nr; j++)
work[j] += p[i].alloc[j];  }  }
if (!flag)       {  printf("No Safe Sequence\n");
return 0;  }  }
printf("Safe Sequence \n");
for (i = 0; i < np; i++)     printf("P%d ", sseq[i]);
printf("\n");   return 1;  }
void newReq()
```

```
{  int pid, i, j, chk1 = 0, chk2 = 0;
printf("Enter Process ID : ");  scanf("%d", &pid);
printf("Enter Request Matrix : \n");
for (j = 0; j < nr; j++)
{  scanf("%d", &req[j]);
if (req[j] > p[pid].need[j])            chk1 = 1;
if (req[j] > avail[j])                  chk2 = 1;   }
if (chk1) {
 printf("Process Exceeds Max Need\n");  return; }
if (chk2)
{  printf("Lack Of Resources\n");  return;  }
for (j = 0; j < nr; j++)
{ avail[j] -= req[j];  p[pid].alloc[j] += req[j];
p[pid].need[j] -= req[j];  }
if (!safe())
{  for (j = 0; j < nr; j++)
{ avail[j] += req[j];  p[pid].alloc[j] -= req[j];
p[pid].need[j] += req[j];  }  }
else       printf("Request Committed\n");  }
void display()  {   int i, j;
printf("Number of Process : %d\n", np);
printf("Number of Resources : %d\n", nr);
printf("PID\tMax\tAllocated\tNeed\n");
for (i = 0; i < np; i++)
{ printf("P%d\t", i);      for (j = 0; j < nr; j++)
printf("%d ", p[i].max[j]);   printf("\t");
for (j = 0; j < nr; j++)      printf("%d ", p[i].alloc[j]);
printf("\t");      for (j = 0; j < nr; j++)
printf("%d ", p[i].need[j]);  printf("\n");  }
printf("Available\n");         for (i = 0; i < nr; i++)
printf("%d ", avail[i]);  printf("\n"); }
void main1()
{   int ch;              for (;;)
{ printf("1)Input 2)NewRequest 3)Safe 4)Display
5)Exit\n");   printf("Enter Choice : ");
scanf("%d", &ch);
switch (ch)
{  case 1:            input();  break;
case 2:               newReq();  break;
case 3:               safe();  break;
case 4:               display();  break;
case 5:               exit(0);  }  }  }
```

3.
```
#include<stdio.h>  #include<string.h>
#include<stdlib.h>  int n,i,j,k,count;
char grm[10][20], fst[10][20],
fol[10][20], tble[3][4], inp[20],
inpt[20],  mch[20], stk[20];
void firstSet();  void followSet();
void parsingTable();  void parseInput();
void print(char* s);
 void main() {
setbuf(stdout,NULL);
printf("The Given Grammar is : \n");
printf("A->aBa\nB-bB|@\n");
printf("Enter The Number Of Rules : ");
scanf("%d", &n);
printf("Enter The Rules : \n");
for (i = 0; i < n; i++)
scanf("%s", grm[i]);  firstSet();
followSet();  parsingTable();
parseInput();  }
void firstSet() {
printf("The First Set Is : \n");
for (i = 0; i < n; i++) {
count = 0;  j = 3;
printf("FIRST[%c]={", grm[i][0]);
while (grm[i][j] != '\0') {
if (!(grm[i][j] >= 65 && grm[i][j] <= 90)) {
fst[i][count++] = grm[i][j];
printf("%c,", grm[i][j]);  }
while (grm[i][j] != '|' && grm[i][j] != '\0')
j++;  j++;  }
printf("\b}\n");  }  }
void followSet() {
printf("The Follow Set Is : \n");
for (k = 0; k < n; k++) {
count = 0;
printf("FOLLOW[%c]={", grm[k][0]);
if (k == 0) {
printf("$,");  fol[k][count++] = '$';  }
for (i = 0; i < n; i++) {
for (j = 3; grm[i][j] != '\0'; j++) {
if (grm[i][j] == grm[k][0] && grm[i][j
+ 1] != '\0'
&& grm[i][j + 1] != '|') {
if (!(grm[i][j + 1] >= 65 && grm[i][j + 1] <=
90)) {
printf("%c,", grm[i][j + 1]);
fol[k][count++] = grm[i][j + 1];  } } } }
printf("\b}\n"); } }
```

```
void parsingTable() {
char p[10], q[10], r[10], f;
strcpy(p, "A->aBa"); strcpy(q, "B->bB");
strcpy(r, "B->@"); tble[1][0] = 'A';
tble[2][0] = 'B'; tble[0][1] = 'a';
tble[0][2] = 'b'; tble[0][3] = '$';
for (i = 0; i < n; i++) {
for (j = 0; fst[i][j] != '\0'; j++) {
f = fst[i][j];
if (f == 'a')
tble[i + 1][1] = 'p';
else if (f == 'b')
tble[i + 1][2] = 'q';
else if (f == '@') {
for (k = 0; fol[i][k] != '\0'; k++)
if (fol[i][k] == 'a')
tble[i + 1][1] = 'r'; } } }
printf("The Parsing Table is : \n");
for (i = 0; i < 3; i++) {
for (j = 0; j < 4; j++) {
if (tble[i][j] == 'p')
printf("%s\t\t", p);
else if (tble[i][j] == 'q')
printf("%s\t\t", q);
else if (tble[i][j] == 'r')
printf("%s\t\t", r);
else     printf("%c\t\t", tble[i][j]); }
printf("\n"); } }
void parseInput() {
printf("Enter The String : ");
scanf("%s", inp);
strcpy(inpt, inp); strcat(inpt, "$");
strcpy(stk, "A$");   i = 0; j = 0; k = 0;
printf("Matched\t\tStack\t\tInput\t\tActio\n";
while (1) { if (stk[i] == inpt[j])
 { if (stk[i] == '$') {
print("Accepted\n"); break; }
print("POP"); printf(" %c\n", stk[i]);
mch[k++] = stk[i]; stk[i++] = inpt[j++] = ' ';
continue; } else if (stk[i] == 'A') {
print("A->aBa\n"); strcpy(stk, "aBa$");
} else if (stk[i] == 'B' && inpt[j] == 'b') {
print("B->bB\n"); strcpy(stk, "bBa$"); i = 0;
} else if (stk[i] == 'B' && inpt[j] == 'a') {
print("B->@\n"); stk[i++] = ' '; } else {
print("ERROR\n"); exit(0); } } }
void print(char* s) {
printf("%s\t\t%s\t\t%s\t\t%s", mch, stk, inpt,
s); }
```