Table of Contents

[2 One 1](#_Toc108730216)

[2.1 1a 1](#_Toc108730217)

[2.2 1b.l 2](#_Toc108730218)

[2.3 1b.y 2](#_Toc108730219)

[3 Two 3](#_Toc108730220)

[3.1 2.l 3](#_Toc108730221)

[3.2 2.y 3](#_Toc108730222)

[4 Three 4](#_Toc108730223)

[4.1 3.c 4](#_Toc108730224)

[5 Four 6](#_Toc108730225)

[5.1 4.c 6](#_Toc108730226)

[6 Five 8](#_Toc108730227)

[6.1 5.c 8](#_Toc108730228)

[7 Six 10](#_Toc108730229)

[7.1 6a.l 10](#_Toc108730230)

[7.2 6b.l 11](#_Toc108730231)

[7.3 6b.y 11](#_Toc108730232)

[8 Seven 13](#_Toc108730233)

[8.1 7.c 13](#_Toc108730234)

[9 Eight 15](#_Toc108730235)

[9.1 8.c 15](#_Toc108730236)

[10 Nine 17](#_Toc108730237)

[10.1 9.c 17](#_Toc108730238)

# One

## 1a

%{

int a[]={0,0}, i, v=1, o=0;

void ex();

%}

%x O

%%

[a-zA-Z0-9]+ { BEGIN O; o++;}

<O>"+" { if(v) { v=0;i=0; } else ex(); }

<O>"\*" { if(v) { v=0;i=1; } else ex(); }

<O>[a-zA-Z0-9]+ { o++;

if(v==0)

{

v=1;a[i]++;

}

else

ex();

}

<O>"\n" { if(v==0)

ex();

else

return 0;

}

.|\n ex();

%%

void ex()

{

printf("invalid expression\n");

exit(0);

}

void main()

{

printf("enter\n");

yylex();

if(v==0)

printf("not valid expression");

else

{

printf("valid expression\n");

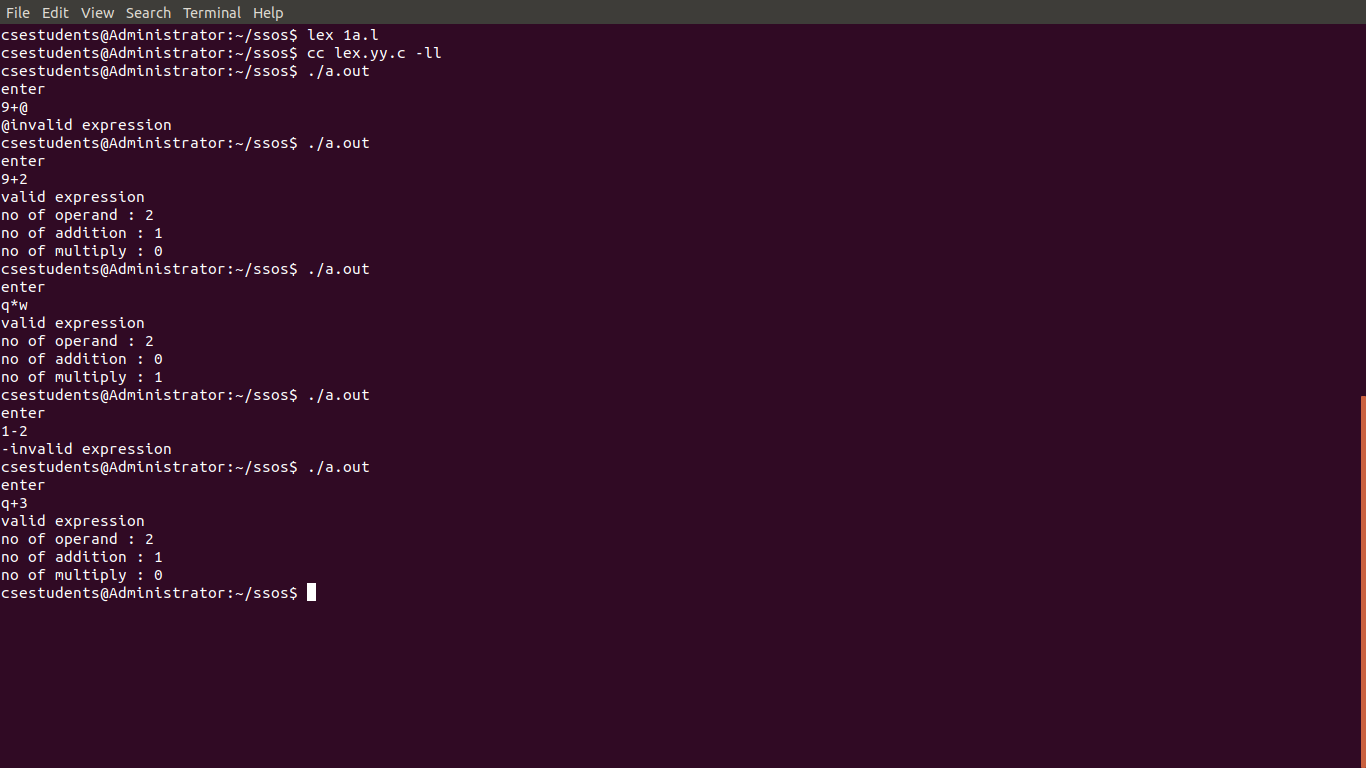
printf("no of operand : %d \n",o);

printf("no of addition : %d \n",a[0]);

printf("no of multiply : %d \n",a[1]);

}

}



## 1b.l

%{

#include "y.tab.h"

extern yylval;

%}

%%

[0-9]+ {yylval=atoi(yytext);return num;}

[\+\-\\*\/] {return yytext[0];}

[)] {return yytext[0];}

[(] {return yytext[0];}

. {;}

\n {return 0;}

%%

## 1b.y

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token num

%left '\*' '/'

%left '+' '-'

%%

input:exp{printf("%d\n",$$);exit(0);}

exp:exp'+'exp {$$=$1+$3;}

|exp'-'exp {$$=$1-$3;}

|exp'\*'exp {$$=$1\*$3;}

|exp'/'exp { if($3==0){printf("divide by 0 error\n");exit(0);}

else

$$=$1/$3;}

|'('exp')' {$$=$2;};

|num{$$=$1;};

%%

int yyerror()

{

printf("error");

exit(0);

}

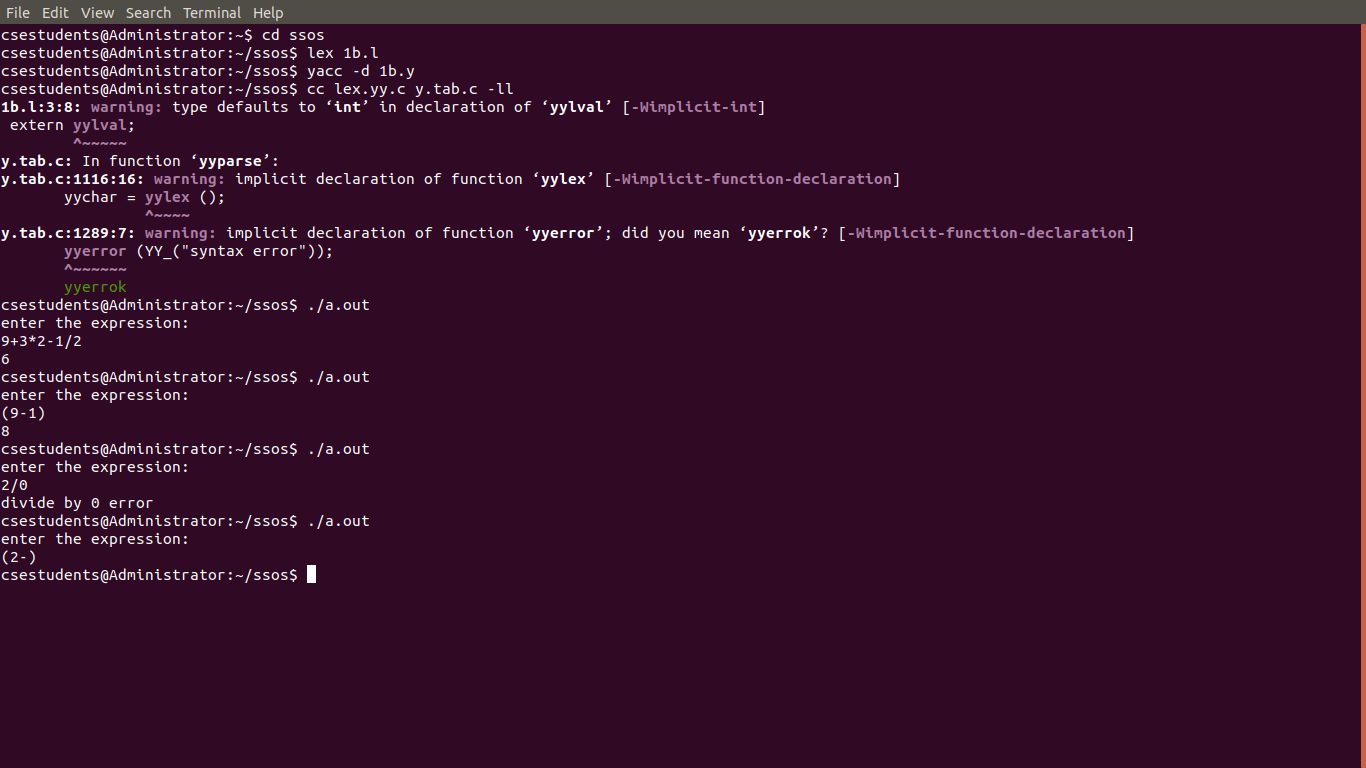
int main()

{

printf("enter the expression:\n");

yyparse();

}



# Two

## 2.l

%{

#include "y.tab.h"

%}

%%

a {return A;}

b {return B;}

[\n] return '\n';

%%

## 2.y

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token A B

%%

input:s'\n' {printf("good grammer\n");exit(0);}

s:A s1 B| B

s1:;|A s1

%%

main()

{

printf("enter a string\n");yyparse();

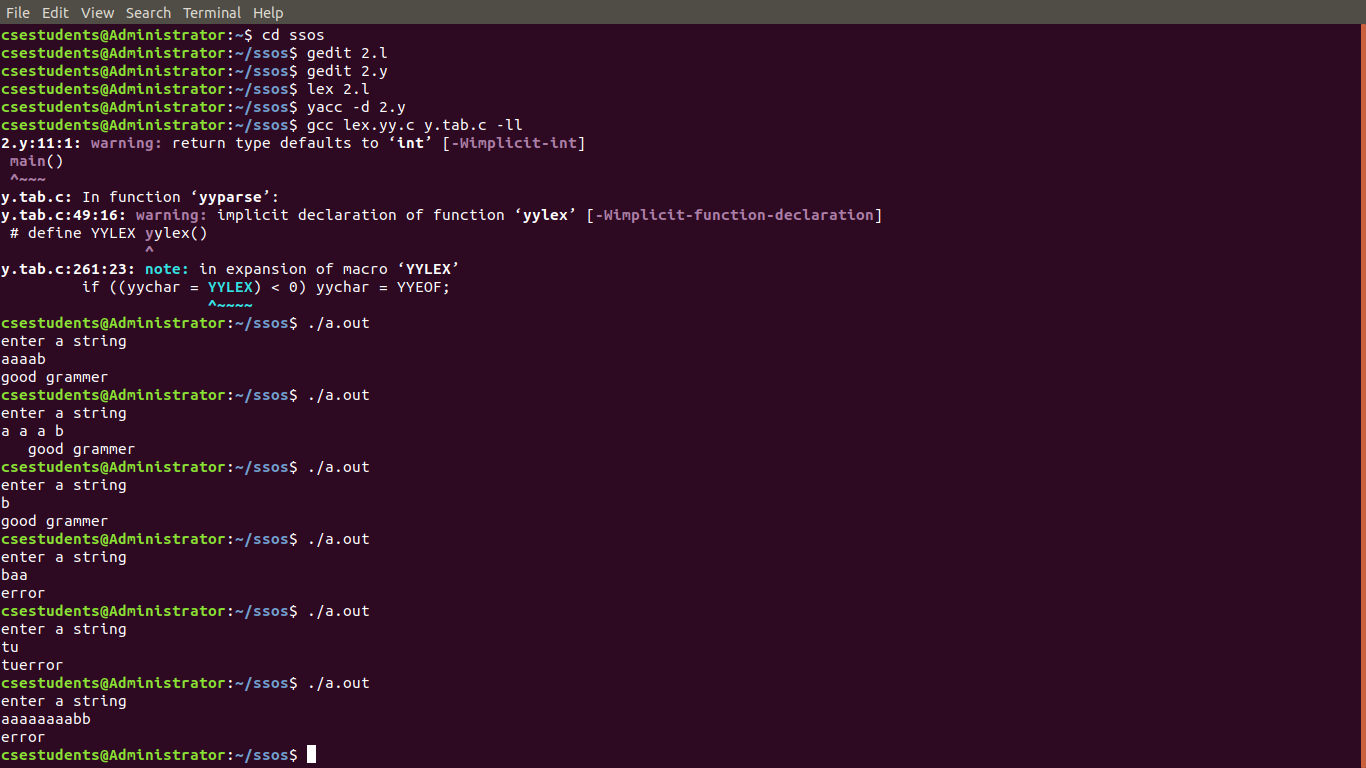
}

int yyerror()

{

printf("error\n");exit(0);

}



# Three

## 3.c

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int num(char c)

{

switch(c)

{

case'A':return 0;

case'B':return 1;

case'a':return 0;

case'b':return 1;

case'@':return 2;

}

return 1;

}

int main()

{

char m[2][3][10]={{"E\0","E\0","E\0"},{"E\0","E\0","E\0"}},ip[100],stack[100];

char first[3][10]={"a\0","b\0","@\0"},follow[3][10]={"$\0","a\0","a\0"},LHS[3][3]={"A\0","B\0","B\0"},

RHS[3][4]={"aBa\0","bB\0","@\0"};

int size[2][3]={3,1,1,1,2,1},p,q,r,i,j,n,k,row,col;

printf("\nfirst={%c,%c,%c}",first[0][0],first[1][0],first[2][0]);

printf("\nfollow={%c,%c}\n\n\n",follow[0][0],follow[1][0]);

for(i=0;i<3;i++)

{

if(first[i][0]!='@')

strcpy(m[num(LHS[i][0])][num(first[i][0])],RHS[i]);

else

strcpy(m[num(LHS[i][0])][num(follow[i][0])],RHS[i]);

}

printf("Input the String:\n");

scanf("%s",ip);

strcat(ip,"$");

n=strlen(ip);

stack[0]='$';

stack[1]='A';

i=1;j=0;

printf("Parsing Table\n");

for(p=0;p<2;p++)

{

for(q=0;q<3;q++)

printf("%s\t",m[p][q]);

printf("\n");

}

printf("\nStack\tInput\n");

for(k=0;k<=i;k++)

printf("%c",stack[k]);

printf("\t");

for(k=j;k<=n;k++)

printf("%c",ip[k]);

printf("\n");

while((stack[i]!='$')&&(ip[j]!='$'))

{

if(stack[i]==ip[j])

{

i--;

j++;

for(k=0;k<=i;k++)

printf("%c",stack[k]);

printf("\t");

for(k=j;k<=n;k++)

printf("%c",ip[k]);

printf("\n");

}

switch(stack[i])

{

case 'A': row=0;break;

case 'B': row=1;break;

default:

if((stack[i]=='$')&&(ip[j]=='$'))

printf("Successful Parsing\n");

else

printf("Parsing Error\n");

exit(0);

}

switch(ip[j])

{

case 'a': col=0; break;

case 'b': col=1; break;

case 'c': col=2; break;

}

if(m[row][col][0]==ip[j])

{

for(k=size[row][col]-1;k>=0;k--)

{

stack[i]=m[row][col][k];

i++;

}

i--;

}

if(m[row][col][0]=='E')

{

if(i>0)

{

printf("Error\n");

exit(0);

}

}

if(m[row][col][0]=='@')

i--;

for(k=0;k<=i;k++)

printf("%c",stack[k]);

printf("\t");

for(k=j;k<=n;k++)

printf("%c",ip[k]);

printf("\n");

}

}



# Four

## 4.c

#include<stdio.h>

#include<string.h>

int k=0,z=0,i=0,j=0,c=0;

char a[16],ac[20],stk[15],act[10];

void check();

void main()

{

puts("GRAMMAR is \nE->E+E \n E->E\*E \n E->(E) \n E->id");

puts("enter input string ");

gets(a);

c=strlen(a);

strcpy(act,"SHIFT->");

puts("\nstack \t input \t action");

for(k=0,i=0; j<c; k++,i++,j++)

{

if(a[j]=='i' && a[j+1]=='d')

{

stk[i]=a[j];

stk[i+1]=a[j+1];

stk[i+2]='\0';

a[j]=' ';

a[j+1]=' ';

printf("$%s\t%s$\t%sid\n",stk,a,act);

check();

}

else

{

stk[i]=a[j];

stk[i+1]='\0';

a[j]=' ';

printf("$%s\t%s$\t%ssymbols\n",stk,a,act);

check();

}

}

}

void check()

{

strcpy(ac,"REDUCE TO E ");

for(z=0; z<c; z++)

if(stk[z]=='i' && stk[z+1]=='d')

{

stk[z]='E';

stk[z+1]='\0';

printf("$%s\t%s$\t%s\n",stk,a,ac);

j++;

}

for(z=0; z<c; z++)

if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')

{

stk[z]='E';

stk[z+1]='\0';

stk[z+2]='\0';

printf("$%s\t%s$\t%s\n",stk,a,ac);

i=i-2;

}

for(z=0; z<c; z++)

if(stk[z]=='E' && stk[z+1]=='\*' && stk[z+2]=='E')

{

stk[z]='E';

stk[z+1]='\0';

stk[z+2]='\0';

printf("$%s\t%s$\t%s\n",stk,a,ac);

i=i-2;

}

for(z=0; z<c; z++)

if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')

{

stk[z]='E';

stk[z+1]='\0';

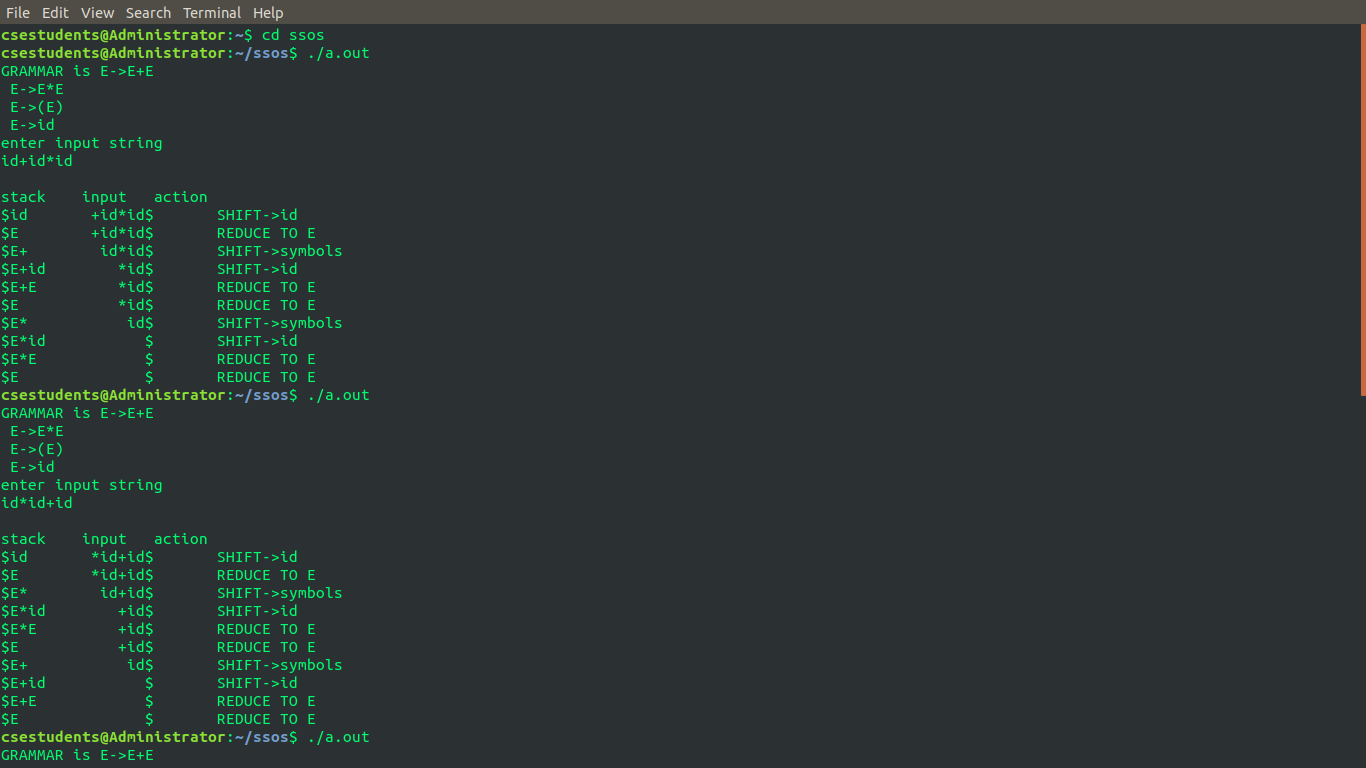
stk[z+2]='\0';

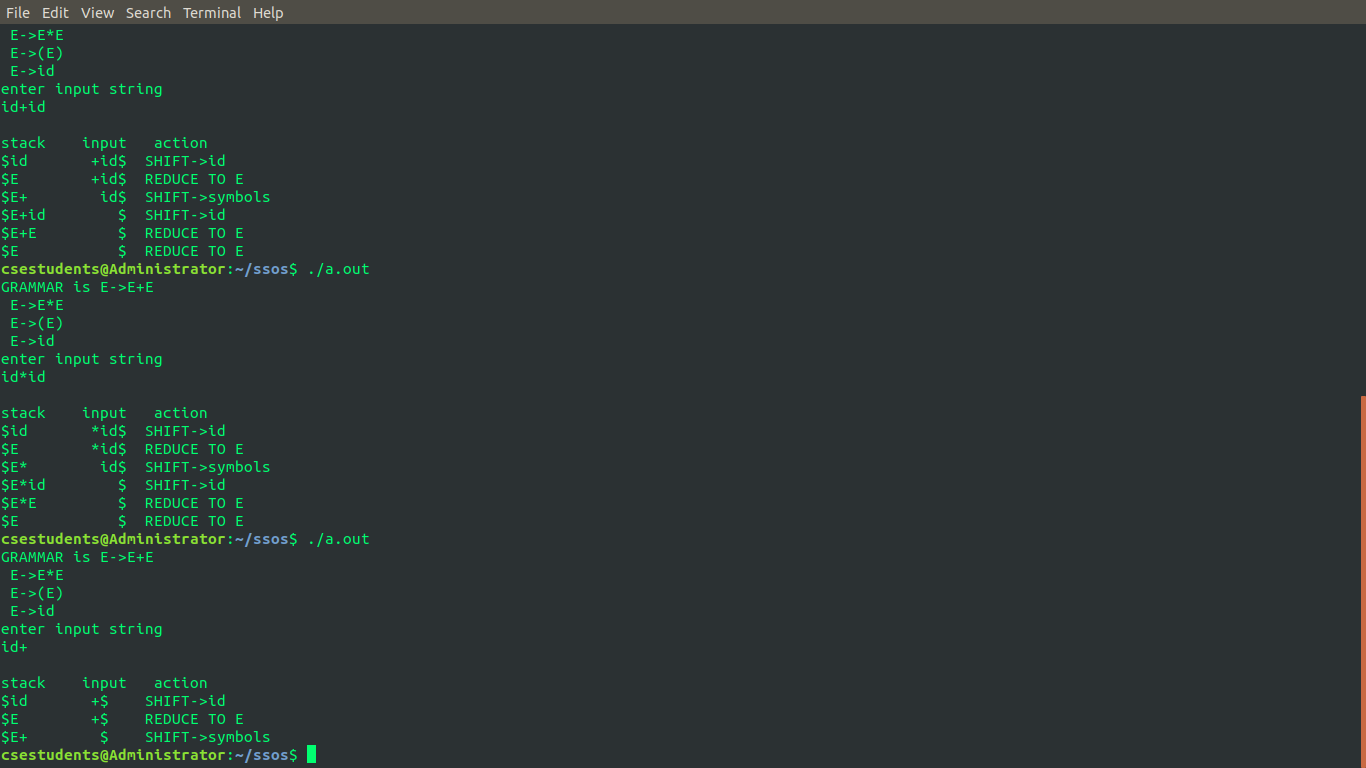
printf("$%s\t%s$\t%s\n",stk,a,ac);

i=i-2;

}

}





# Five

## 5.c

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

char op[2],arg1[5],arg2[5],result[5];

void main()

{

FILE \*fp1,\*fp2;

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

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

while(!feof(fp1))

{

fscanf(fp1,"%s%s%s%s",result,arg1,op,arg2);

if(strcmp(op,"+")==0)

{

fprintf(fp2,"MOV R0,%s\n",arg1);

fprintf(fp2,"ADD R0,%s\n",arg2);

fprintf(fp2,"MOV %s,R0\n",result);

}

if(strcmp(op,"\*")==0)

{

fprintf(fp2,"MOV R0,%s\n",arg1);

fprintf(fp2,"MUL R0,%s\n",arg2);

fprintf(fp2,"MOV %s,R0\n",result);

}

if(strcmp(op,"-")==0)

{

fprintf(fp2,"MOV R0,%s\n",arg1);

fprintf(fp2,"SUB R0,%s\n",arg2);

fprintf(fp2,"MOV %s,R0\n",result);

}

if(strcmp(op,"/")==0)

{

fprintf(fp2,"MOV R0,%s\n",arg1);

fprintf(fp2,"DIV R0,%s\n",arg2);

fprintf(fp2,"MOV %s,R0\n",result);

}

if(strcmp(op,"=")==0)

{

fprintf(fp2,"MOV R0,%s\n",arg1);

fprintf(fp2,"MOV %s,R0\n",result);

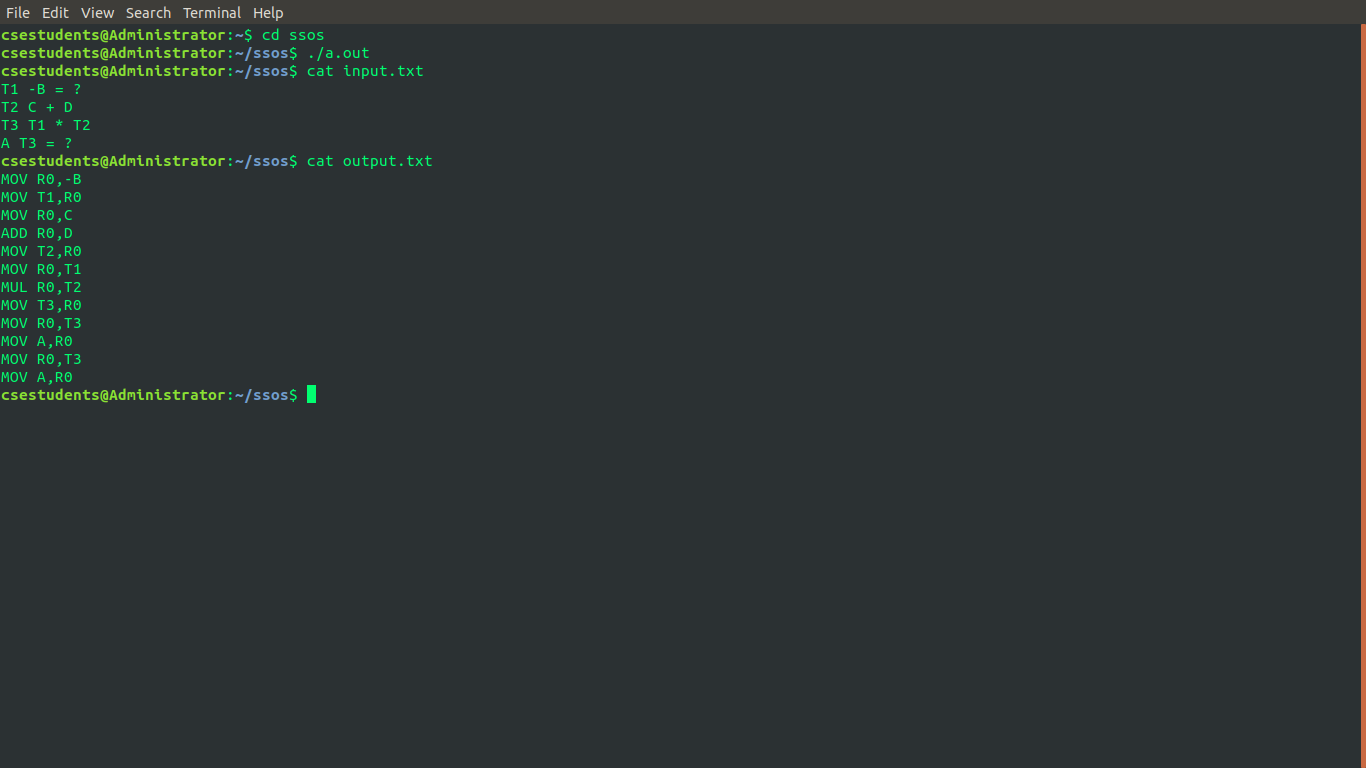
}

}

fclose(fp1);

fclose(fp2);

}



# Six

## 6a.l

%{

#include<stdio.h>

int sl=0;

int ml=0;

%}

%%

"/\*"[^"\*/"]\*"\*/" ml++;

"//".\*[\n] sl++;

%%

main()

{

yyin=fopen("f1.c","r");

yyout=fopen("f2.c","w");

yylex();

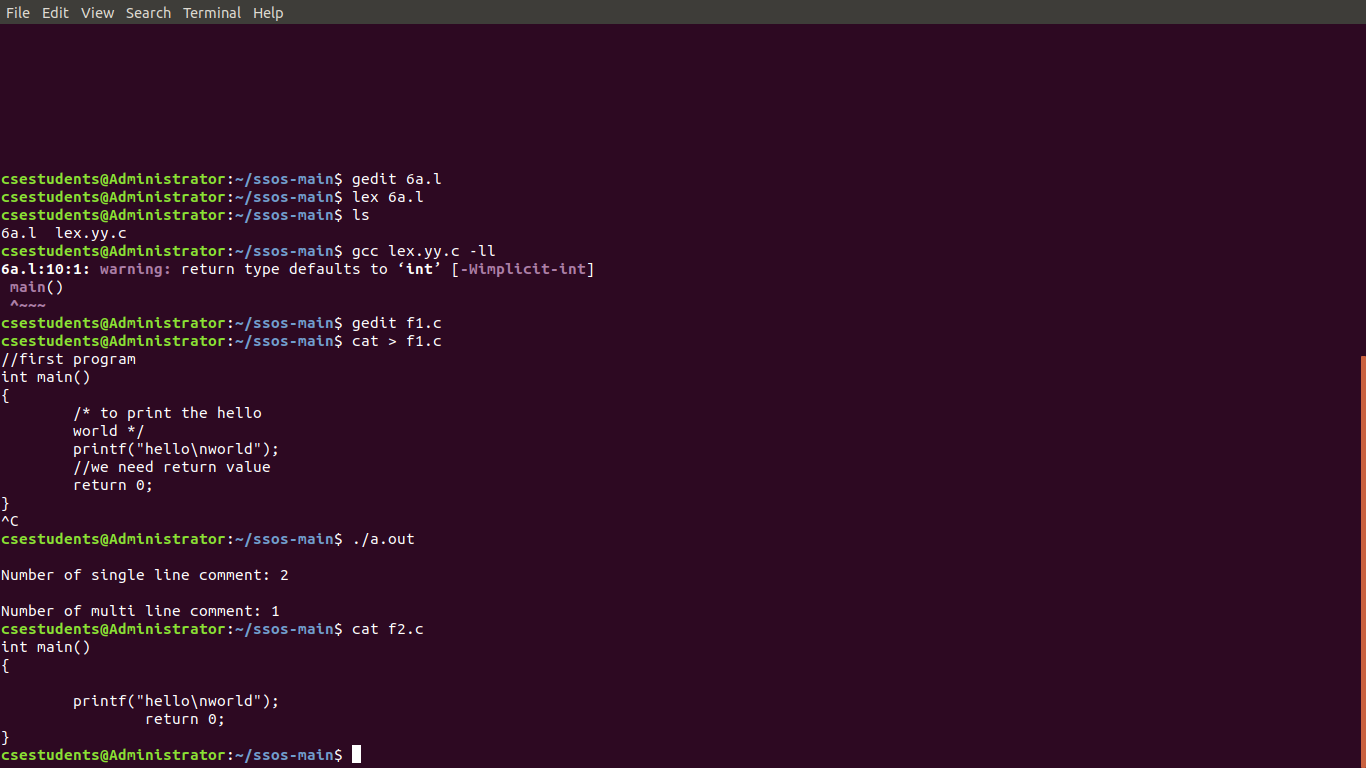
fclose(yyin);

fclose(yyout);

printf("\nNumber of single line comment: %d\n",sl);

printf("\nNumber of multi line comment: %d\n",ml);

}



## 6b.l

%{

#include<stdio.h>

#include"y.tab.h"

extern yylval;

%}

%%

[\t] ;

[+|-|\*|/|+|>|>] {printf("operator is %s\n",yytext);return OP;}

[0-9]+ {yylval=atoi(yytext); printf("numers is %d\n",yylval);return DIGIT;}

int|char|float|void|for|do|while|if|else|return|switch|case {printf("keyword is %s\n",yytext);return KEY;}

[a-zA-Z0-9]+ {printf("identifier is %s\n",yytext);return ID;}

. ;

%%

## 6b.y

%{

#include<stdio.h>

#include<stdlib.h>

int id=0, dig=0, key=0, op=0;

%}

%token DIGIT OP KEY ID

%%

input:

DIGIT input {dig++;}

|ID input {id++;}

|KEY input {key++;}

|OP input {op++;}

|DIGIT {dig++;}

|ID {id++;}

|KEY {key++;}

|OP {op++;}

;

%%

#include<stdio.h>

extern int yylex();

extern int yyparse();

extern FILE \*yyin;

main()

{

FILE \*mf=fopen("f1.c","r");

if(!mf)

{ ("cant open file");

return -1;

}

yyin=mf;

do{

yyparse();

}while(!feof(yyin));

printf("numbers=%d\nkeywords=%d\nidentifiers=%d\noperators=%d\n",dig,key,id,op);

}

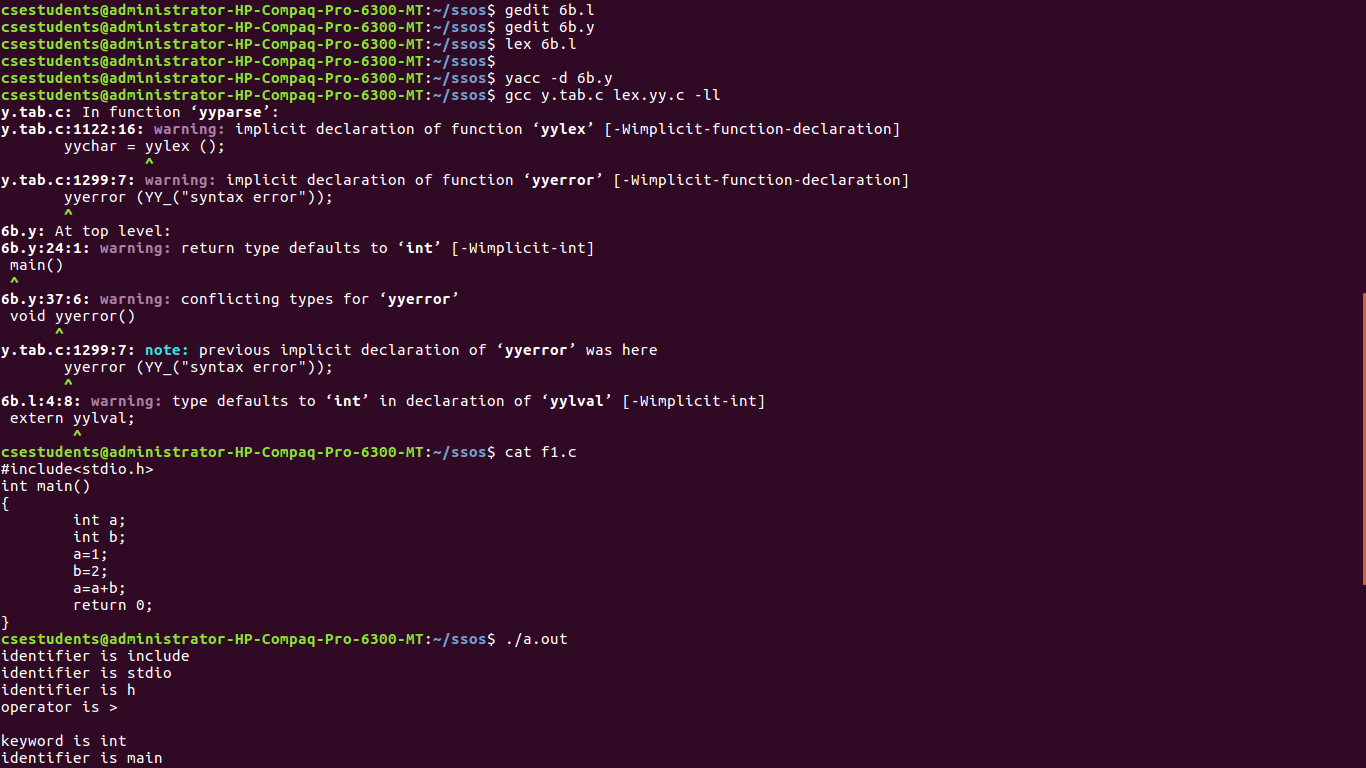
void yyerror()

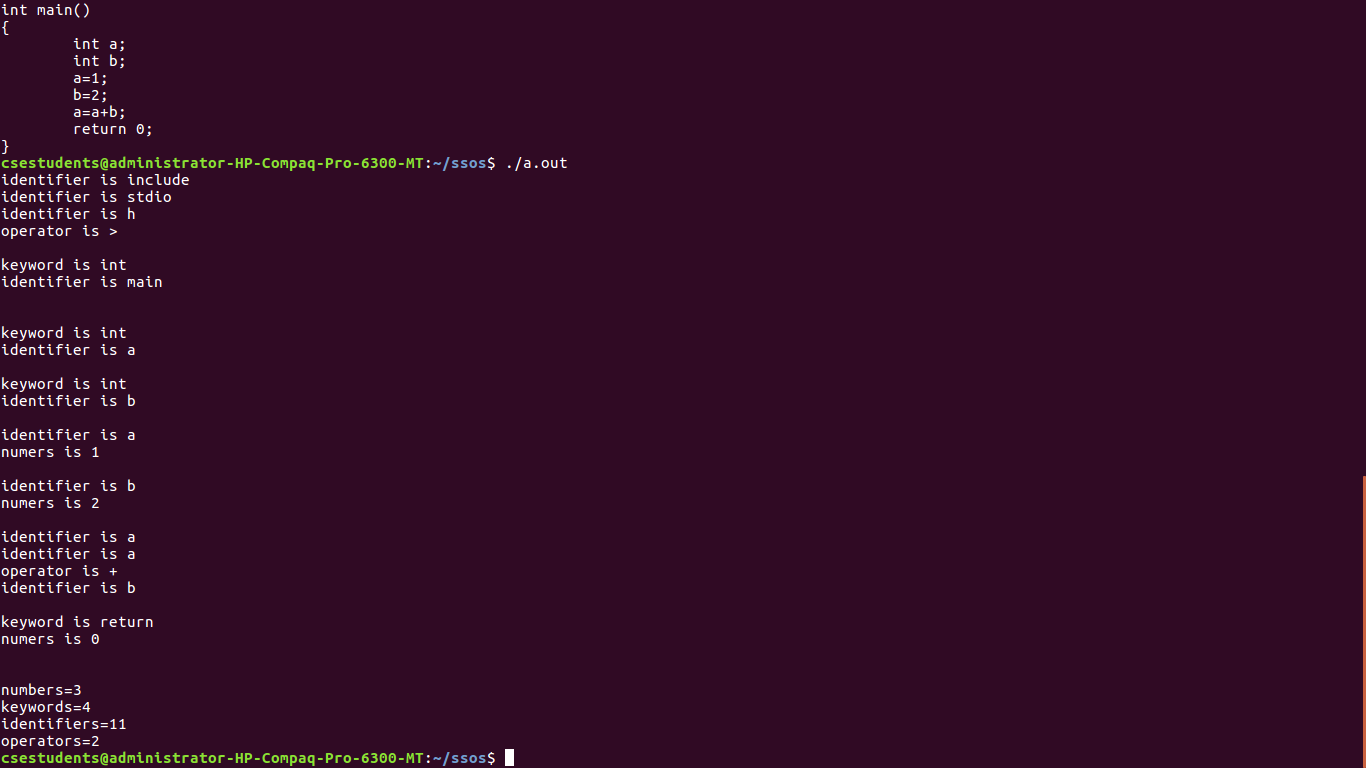
{

printf("error! message: ");

exit(-1);

}





# Seven

## 7.c

#include<stdio.h>

#include<stdlib.h>

typedef struct J

{

int arrival,finish,burst,tat,wt;

}Job;

void scheduler(Job job[],int n,int q,int c)

{

int bursts[100];

for(int i=0;i<n;i++)

bursts[i] = job[i].burst;

int t = 0,done = 0, curr, diff;

float tat\_sum = 0,wt\_sum = 0;

if (c==0)

curr = -1;

else

curr = 0;

while (done<n)

{

if(c==1)

{

for(int x=0;x<n;x++)

{

if(job[curr].burst == 0)

curr = x;

if(job[x].burst < job[curr].burst)

if(job[x].burst > 0 && job[x].arrival <= t)

curr = x;

}

diff = 1;

}

else

{

while(1)

{

curr = (curr + 1) % n;

if(job[curr].burst != 0)

break;

}

diff = (q<=job[curr].burst)?q:job[curr].burst;

}

job[curr].burst -= diff;

t += diff;

if(job[curr].burst == 0)

{

done++;

job[curr].finish = t;

}

}

if(c==1)

printf("\nThe SJF schedule details are\n");

else

printf("\nThe Round Robin Schedule details are\n");

for (int i=0;i<n;i++)

job[i].burst = bursts[i];

printf("\nJob\tTaT\tWT\n");

for(int i=0;i<n;i++)

{

job[i].tat = job[i].finish - job[i].arrival\*c;

job[i].wt = job[i].tat - job[i].burst;

printf("%d\t%d\t%d\n",i+1 ,job[i].tat,job[i].wt);

tat\_sum += job[i].tat;

wt\_sum += job[i].wt;

}

printf("\nAvg Turnaround Time = %f\nAvg Waiting Time = %f\n",tat\_sum/n, wt\_sum/n);

}

int main()

{

Job job[100];

int n,q,c;

printf("Enter the number of jobs\n");

scanf("%d", &n);

printf("Enter Arrival Burst\n");

for(int i=0;i<n;i++)

{

printf("J%d: ",i+1);

scanf("%d%d", &job[i].arrival, &job[i].burst);

}

printf("1:Round Robin\n2:Shortest Job First\n");

scanf("%d",&c);

switch (c)

{

case 1:

printf("Enter quantum for Round Robin\n");

scanf("%d",&q);

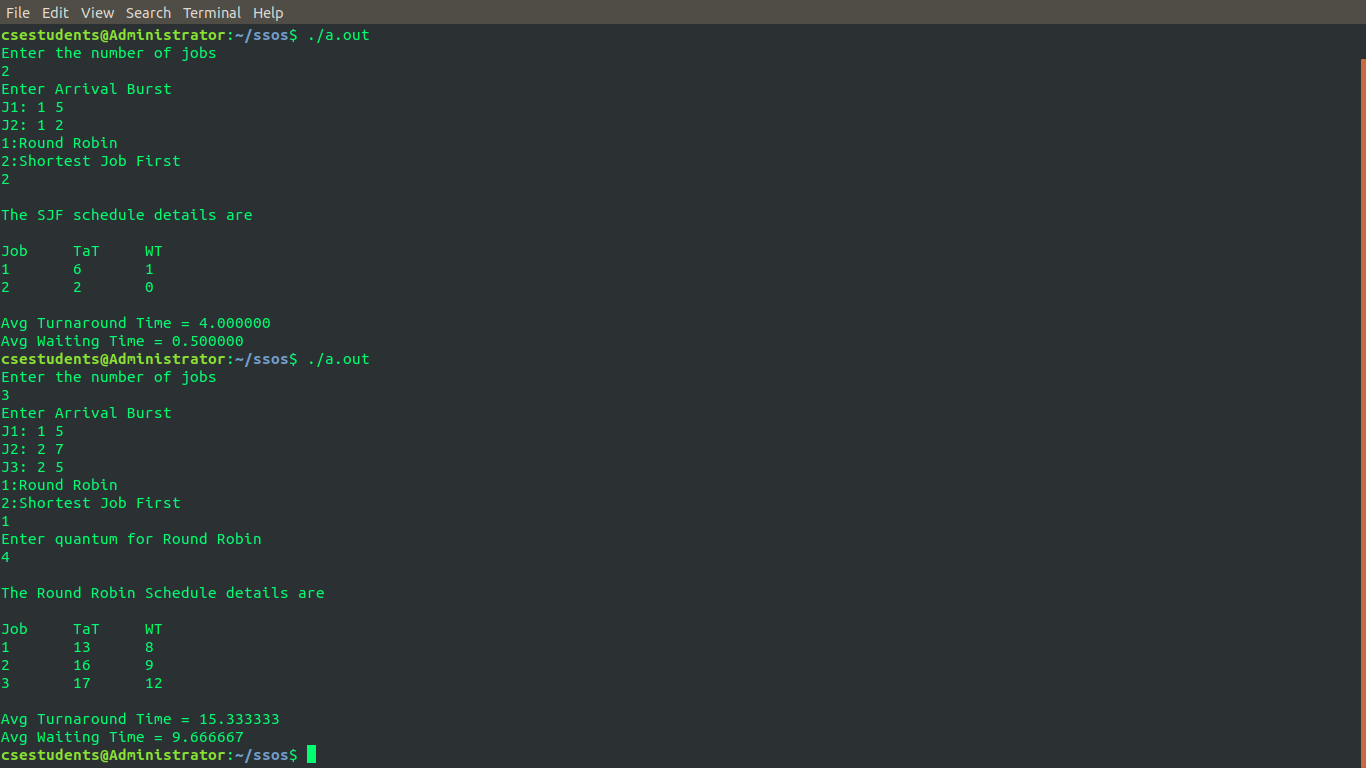
scheduler(job, n, q, 0);

break;

case 2:scheduler(job, n, q, 1);

}

}



# Eight

## 8.c

#include <stdio.h>

#include <stdlib.h>

int main()

{

int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];

int p, r, i, j, process, count;

count = 0;

printf("Enter the no of processes : ");

scanf("%d", &p);

for(i = 0; i< p; i++)

completed[i] = 0;

printf("Enter the no of resources : ");

scanf("%d", &r);

printf("Enter the Max Matrix for each process : \n");

for(i = 0; i < p; i++)

{

printf("For process %d : ", i + 1);

for(j = 0; j < r; j++)

scanf("%d", &Max[i][j]);

}

printf("Enter the allocation for each process : \n");

for(i = 0; i < p; i++)

{

printf("For process %d : ",i + 1);

for(j = 0; j < r; j++)

scanf("%d", &alloc[i][j]);

}

printf("Enter the Available Resources : ");

for(i = 0; i < r; i++)

scanf("%d", &avail[i]);

for(i = 0; i < p; i++)

for(j = 0; j < r; j++)

need[i][j] = Max[i][j] - alloc[i][j];

do

{

printf("Max matrix:\t\tAllocation matrix:\n");

for(i = 0; i < p; i++)

{

for( j = 0; j < r; j++)

printf("%d ", Max[i][j]);

printf("\t\t");

for( j = 0; j < r; j++)

printf("%d ", alloc[i][j]);

printf("\n");

}

process = -1;

for(i = 0; i < p; i++)

{

if(completed[i] == 0)//if not completed

{

process = i ;

for(j = 0; j < r; j++)

{

if(avail[j] < need[i][j])

{

process = -1;

break;

}

}

}

if(process != -1)

break;

}

if(process != -1)

{

printf("Process %d runs to completion!\n", process + 1);

safeSequence[count] = process + 1;

count++;

for(j = 0; j < r; j++)

{

avail[j] += alloc[process][j];

alloc[process][j] = 0;

Max[process][j] = 0;

completed[process] = 1;

}

}

}

while(count != p && process != -1);

if(count == p)

{

printf("The system is in a safe state!!\n");

printf("Safe Sequence : < ");

for( i = 0; i < p; i++)

printf("%d ", safeSequence[i]);

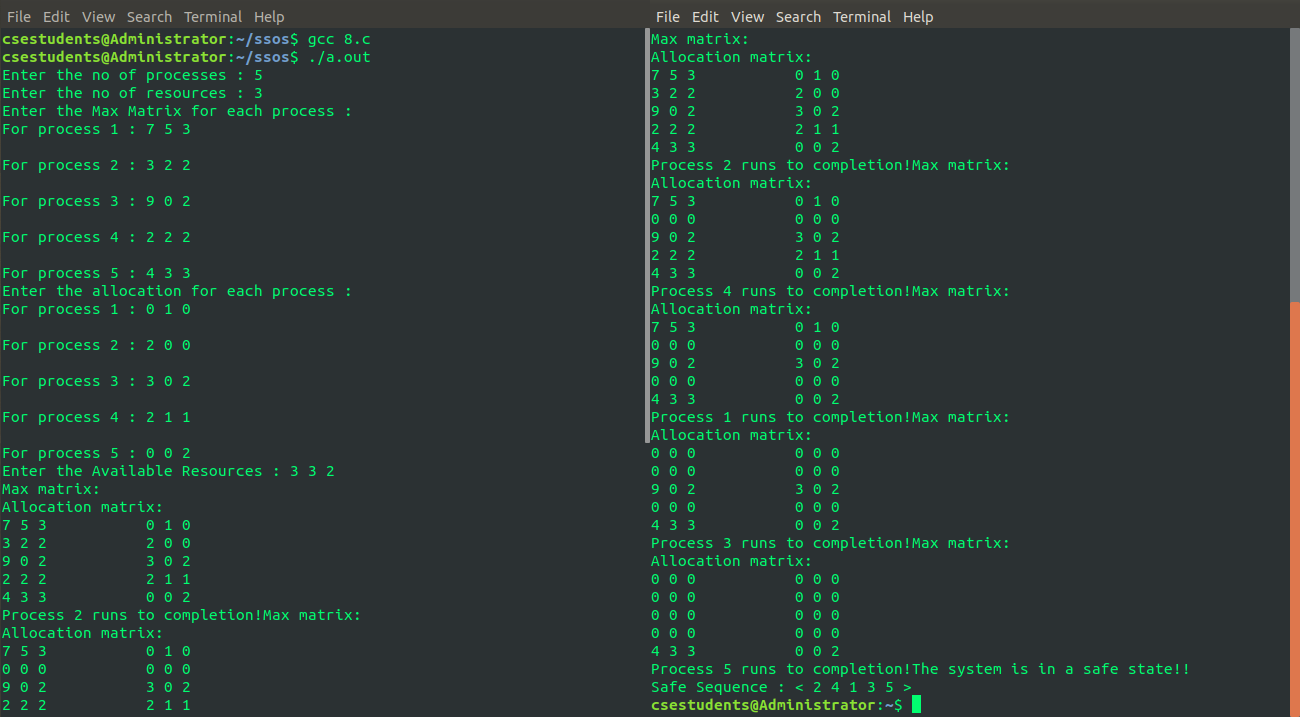
printf(">\n");

}

else

printf("The system is in an unsafe state!!");

}



# Nine

## 9.c

#include<stdio.h>

#include<stdlib.h>

void FIFO(char s[],char F[],int l,int f)

{

int i,j=0,k,flag=0;

printf("PAGE\tFRAMES\tFAULTS");

for(i=0;i<l;i++)

{

for(k=0;k<f;k++)

if(F[k]==s[i])

flag=1;

printf("\n%c\t",s[i]);

if(flag==0)

{

F[j++]=s[i];

printf("%s",F);

printf("\tPage Fault");

}

else

{

flag=0;

printf("%s",F);

printf("\tPage Hit");

}

if(j==f)

j=0;

}

}

void lru(char s[],char F[],int l,int f)

{

int i,j=0,k,m,flag=0,top=0;

printf("\nPAGE\t FRAMES\t FAULTS");

for(i=0;i<l;i++)

{

for(k=0;k<f;k++)

if(F[k]==s[i])

flag=1;

printf("\n%c\t",s[i]);

if(j!=f && flag!=1)

{

F[top]=s[i];

if(++j!=f)

top++;

}

else

{

if(flag!=1)

{

for(k=0;k<top;k++)

F[k]=F[k+1];

F[top]=s[i];

}

else

{

for(m=k;m<top;m++)

F[m]=F[m+1];

F[top]=s[i];

}

}

printf("%s",F);

if(flag==0)

printf("\tPage Fault");

else

printf("\tPage Hit");

flag=0;

}

}

void main()

{

int ch,i,l,f;

char F[10],s[25];

printf("Enter the no. of frames: ");

scanf("%d",&f);

F[f]='\0';

printf("Enter the length of the string: ");

scanf("%d",&l);

printf("Enter the string: ");

scanf("%s",s);

while(1)

{

printf("\nEnter:\n1:FIFO\n2:LRU\n3:EXIT\n");

scanf("%d",&ch);

switch(ch)

{

case 1: for(i=0;i<f;i++)

F[i]=-1;

FIFO(s,F,l,f);

break;

case 2: for(i=0;i<f;i++)

F[i]=-1;

lru(s,F,l,f);

break;

case 3: exit(0);

}

}

}

