**Pass 1**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void main(){

FILE \*input, \*optab, \*symtab,\*intermediate,\*prgmlength;

int locctr, start, flag, program\_length;

char opcode[20], operand[20], label[20], mneumonic[20], code[20];

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

optab = fopen("optab.txt", "r");

symtab = fopen("symtab.txt", "w");

intermediate = fopen("intermediate.txt", "w");

prgmlength=fopen("prgmlength.txt","w");

if (input == NULL || optab == NULL || symtab == NULL || intermediate == NULL) {

printf("Error opening file!\n");

exit(0);

}

fscanf(input, "%s %s %s", label, opcode, operand);

if (strcmp(opcode, "START") == 0) {

start = atoi(operand);

locctr = start;

fprintf(intermediate, " %-7s%-7s%-7s\n",label,opcode, operand);

}

else {

locctr = 0;

}

printf("LOCCTR:\n%d\n",locctr);

fscanf(input, "%s %s %s", label, opcode, operand);

while (strcmp(opcode, "END") != 0) {

fprintf(intermediate, "%d %-7s%-7s%-7s\n", locctr, label, opcode, operand);

flag = 0;

if (strcmp(label, "\*\*") != 0) {

fprintf(symtab, "%-7s%-7d\n", label, locctr);

}

if (strcmp(opcode, "WORD") == 0) {

locctr += 3;

}

else if (strcmp(opcode, "RESW") == 0) {

locctr += (3 \* atoi(operand));

}

else if (strcmp(opcode, "RESB") == 0) {

locctr += atoi(operand);

}

else if (strcmp(opcode, "BYTE") == 0) {

locctr += (strlen(operand)-3);

}

else {

locctr += 3;

flag = 1;

}

printf("%d\n",locctr);

fscanf(input, "%s %s %s", label, opcode, operand);

}

fprintf(intermediate, "%d %-7s%-7s%-7s\n",locctr,label, opcode, operand);

program\_length = locctr - start;

printf("%d\n", program\_length);

fprintf(prgmlength,"%d",program\_length);

fclose(input);

fclose(intermediate);

fclose(symtab);

fclose(optab);

fclose(prgmlength);

}

**Pass 2**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void main() {

FILE \*intermediate, \*optab, \*symtab, \*output, \*prgmlength,\*objectcode;

char opcode[20], operand[20], label[20], mneumonic[20], code[20], value[20], textRecord[70];

int locctr, start, programLength, length = 0, textStartAddr;

optab = fopen("optab.txt", "r");

intermediate = fopen("intermediate.txt", "r");

symtab = fopen("symtab.txt", "r");

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

objectcode=fopen("objectcode.txt","w");

prgmlength = fopen("prgmlength.txt", "r");

if (!optab || !intermediate || !symtab || !output || !prgmlength) {//to check whether there is any error in opening the file

printf("Error opening files\n");

return;

}

fscanf(intermediate, "%s %s %s", label, opcode, operand);//to read 1st input line

printf("first input line read: %s %s %s\n", label, opcode, operand);

fprintf(output, " %-7s%-7s%-7s\n", label, opcode, operand);//write the listing line to an output file

if (strcmp(opcode, "START") == 0) {

fscanf(prgmlength, "%d", &programLength);

start = atoi(operand);//atoi()is used to convert string to an integer

fprintf(objectcode, "H^%s^%06d^%06d\n", label, start, programLength); // header format

}

fscanf(intermediate, "%d %s %s %s", &locctr, label, opcode, operand);

textStartAddr = locctr;//textStartAddr contains the starting address of a text record

strcpy(textRecord, "");//initialising a text record with a null terminator so that it can be used for other operations

while (strcmp(opcode, "END") != 0) {

char objectCode[10] = "";//initialising objectCode with a null terminator \0

printf("Processing line: %d %s %s %s\n", locctr, label, opcode, operand);

fprintf(output, "%d %-7s%-7s%-7s", locctr, label, opcode, operand);

if (strcmp(opcode, "BYTE") == 0) {

if (operand[0] == 'C' && operand[1] == '\'') {//Checks if the first character of the operand string is C and second character is a single quote '.

if (strcmp(operand, "C'EOF'") == 0) {

strcpy(objectCode, "454F46");

} else {

for (int i = 2; i < strlen(operand) - 1; i++) {

//sprintf() function in C is used to write formatted data to a string or buffer(here to the location pointed by objectCode + strlen(objectCode) which is null \0)

sprintf(objectCode + strlen(objectCode), "%02X", operand[i]);

}

}

}

/\*while (strlen(objectCode) < 6) {//insert 0's to the left if the no of digits in objectcode is less than 6

char temp[10];

sprintf(temp, "0%s", objectCode);

strcpy(objectCode, temp);

}\*/

} else if (strcmp(opcode, "WORD") == 0) {

sprintf(objectCode, "%06d", atoi(operand));

} else if (strcmp(opcode, "RESB") == 0) {

// No object code for RESB

} else if (strcmp(opcode, "RESW") == 0) {

// No object code for RESW

} else {

rewind(optab);//reset the file pointer of the file optab to the beginning of the file

while (fscanf(optab, "%s %s", mneumonic, code) != EOF) {

if (strcmp(opcode, mneumonic) == 0) {//read pairs of strings from the file optab.txt until the end of the file(EOF) is reached.

strcpy(objectCode, code);//If a match is found,the next line copies the code to objectCode, and the loop breaks

break;

}

}

rewind(symtab);

while (fscanf(symtab, "%s %s", label, value) != EOF) {

if (strcmp(operand, label) == 0) {

sprintf(objectCode + strlen(objectCode), "%04d", atoi(value));

break;

}

}

}

printf("Object Code: %s\n", objectCode);

fprintf(output, "%s\n",objectCode);

if (length + strlen(objectCode) > 60) {//to check whether the text record is full

fprintf(objectcode, "T^%06d^%02X^%s\n", textStartAddr, length / 2, textRecord);

strcpy(textRecord, "");//initialising new text record

textStartAddr = locctr;

length = 0;

}

if (strlen(objectCode) > 0) {//adding object code to text record

strcat(textRecord, objectCode);

strcat(textRecord, "^");

length += strlen(objectCode);

}

fscanf(intermediate, "%d %s %s %s", &locctr, label, opcode, operand);//reading next input line

}

if (length > 0) {

textRecord[strlen(textRecord) - 1] = '\0'; // Remove the last '^'

fprintf(objectcode, "T^%06d^%02X^%s\n", textStartAddr, length / 2, textRecord); // Ensure the last text record is written

}

fprintf(output, "%d %-7s%-7s%-7s", locctr, label, opcode, operand);

fprintf(objectcode, "E^%06d\n", start);

fclose(intermediate);

fclose(symtab);

fclose(optab);

fclose(output);

fclose(prgmlength);

printf("FINISHED EXECUTION!!!\n");

}

**CPU Scheduling**

#include<stdio.h>

void FCFS();

void SJF();

void Priority();

void RR();

void main(){

int choice;

while(choice!=5){

printf("\n--->1.FCFS 2.SJF 3.priority 4.RR 5.EXIT\nenter the choice:");

scanf("%d",&choice);

switch(choice){

case 1:{

FCFS();

break;

}

case 2:{

SJF();

break;

}

case 3:{

Priority();

break;

}

case 4:{

RR();

break;

}

case 5:{

printf("EXITED!!\n");

break;

}

}

}

}

void FCFS(){

int limit,tot\_wt=0,tot\_tat=0;

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

scanf("%d",&limit);

int process[limit],bt[limit],wt[limit],tat[limit];

printf("Enter the process and the corresponding burst time:");

for(int i=0;i<limit;i++){

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

}

wt[0]=0;

tat[0]=bt[0];

printf("PROCESS BURST\_TIME TURN\_AROUND\_TIME WAITING\_TIME\n");

printf(" %d %d %d %d\n",process[0],bt[0],tat[0],wt[0]);

tot\_tat=tot\_tat+bt[0];

for(int i=1;i<limit;i++){

wt[i]=wt[i-1]+bt[i-1];

tat[i]=tat[i-1]+bt[i];

printf(" %d %d %d %d\n",process[i],bt[i],tat[i],wt[i]);

tot\_wt=tot\_wt+wt[i];

tot\_tat=tot\_tat+tat[i];

}

printf("GANTT CHART:\n");

for(int i=0;i<limit;i++){

printf(" |P%d| ",process[i]);

}

printf("\n");

for(int i=0;i<limit;i++){

if(i==0){

printf("0 %d",tat[i]);

}

else{

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

}

}

printf("\n");

printf("Total turn around time=%d\n",tot\_tat);

printf("Total waitng time=%d\n",tot\_wt);

printf("average turn around time=%f\n",((float)tot\_tat/limit));

printf("average waitng time=%f\n",((float)tot\_wt/limit));

}

void SJF(){

int limit,tot\_wt=0,tot\_tat=0,temp,temp1;

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

scanf("%d",&limit);

int process[limit],bt[limit],wt[limit],tat[limit];

printf("Enter the process and the corresponding burst time:");

for(int i=0;i<limit;i++){

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

}

for(int i=0;i<limit;i++){

for(int j=i+1;j<limit;j++){

if(bt[i]>bt[j]){

temp=bt[i];

bt[i]=bt[j];

bt[j]=temp;

temp1=process[i];

process[i]=process[j];

process[j]=temp1;

}

}

}

wt[0]=0;

tat[0]=bt[0];

printf("PROCESS BURST\_TIME TURN\_AROUND\_TIME WAITING\_TIME\n");

printf(" %d %d %d %d\n",process[0],bt[0],tat[0],wt[0]);

tot\_tat=tot\_tat+bt[0];

for(int i=1;i<limit;i++){

wt[i]=wt[i-1]+bt[i-1];

tat[i]=tat[i-1]+bt[i];

printf(" %d %d %d %d\n",process[i],bt[i],tat[i],wt[i]);

tot\_wt=tot\_wt+wt[i];

tot\_tat=tot\_tat+tat[i];

}

printf("GANTT CHART:\n");

for(int i=0;i<limit;i++){

printf(" |P%d| ",process[i]);

}

printf("\n");

for(int i=0;i<limit;i++){

if(i==0){

printf("0 %d",tat[i]);

}

else{

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

}

}

printf("\n");

printf("Total turn around time=%d\n",tot\_tat);

printf("Total waitng time=%d\n",tot\_wt);

printf("average turn around time=%f\n",((float)tot\_tat/limit));

printf("average waitng time=%f\n",((float)tot\_wt/limit));

}

void Priority(){

int limit,tot\_wt=0,tot\_tat=0,temp,temp1,temp2;

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

scanf("%d",&limit);

int process[limit],bt[limit],wt[limit],tat[limit],priority[limit];

printf("Enter the process and the corresponding burst time and priority:");

for(int i=0;i<limit;i++){

scanf("%d%d%d",&process[i],&bt[i],&priority[i]);

}

for(int i=0;i<limit;i++){

for(int j=i+1;j<limit;j++){

if(priority[i]>priority[j]){

temp=bt[i];

bt[i]=bt[j];

bt[j]=temp;

temp1=process[i];

process[i]=process[j];

process[j]=temp1;

temp2=priority[i];

priority[i]=priority[j];

priority[j]=temp2;

}

}

}

wt[0]=0;

tat[0]=bt[0];

printf("PROCESS BURST\_TIME PRIORITY TURN\_AROUND\_TIME WAITING\_TIME\n");

printf(" %d %d %d %d %d\n",process[0],bt[0],priority[0],tat[0],wt[0]);

tot\_tat=tot\_tat+bt[0];

for(int i=1;i<limit;i++){

wt[i]=wt[i-1]+bt[i-1];

tat[i]=tat[i-1]+bt[i];

printf(" %d %d %d %d %d\n",process[i],bt[i],priority[i],tat[i],wt[i]);

tot\_wt=tot\_wt+wt[i];

tot\_tat=tot\_tat+tat[i];

}

printf("GANTT CHART:\n");

for(int i=0;i<limit;i++){

printf(" |P%d| ",process[i]);

}

printf("\n");

for(int i=0;i<limit;i++){

if(i==0){

printf("0 %d",tat[i]);

}

else{

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

}

}

printf("\n");

printf("Total turn around time=%d\n",tot\_tat);

printf("Total waitng time=%d\n",tot\_wt);

printf("average turn around time=%f\n",((float)tot\_tat/limit));

printf("average waitng time=%f\n",((float)tot\_wt/limit));

}

void RR(){

int limit,tot\_wt=0,tot\_tat=0,tq,total=0,flag=0,i,q[50];

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

scanf("%d",&limit);

int process[limit],bt[limit],wt[limit],tat[limit],rt[limit];

printf("Enter the process and the corresponding burst time:");

for(int i=0;i<limit;i++){

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

rt[i]=bt[i];

}

printf("Enter the time quantum:");

scanf("%d",&tq);

int a=0;

q[a]=0;

while(flag!=limit){

for(i=0;i<limit;i++){

if(rt[i]>tq){

total=total+tq;

rt[i]=rt[i]-tq;

q[++a]=q[a]+tq;

printf(" |P%d| ",process[i]);

}

else if(rt[i]==0){

continue;

}

else{

total=total+rt[i];

q[++a]=q[a]+rt[i];

rt[i]=0;

tat[i]=total;

wt[i]=total-bt[i];

printf(" |P%d| ",process[i]);

flag++;

}

}

}

printf("\n");

for(int i=0;i<=a;i++){

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

}

printf("\nPROCESS BURST\_TIME TURN\_AROUND\_TIME WAITING\_TIME\n");

for(int i=0;i<limit;i++){

printf(" %d %d %d %d\n",process[i],bt[i],tat[i],wt[i]);

tot\_wt=tot\_wt+wt[i];

tot\_tat=tot\_tat+tat[i];

}

printf("\n");

printf("Total turn around time=%d\n",tot\_tat);

printf("Total waitng time=%d\n",tot\_wt);

printf("average turn around time=%f\n",((float)tot\_tat/limit));

printf("average waitng time=%f\n",((float)tot\_wt/limit));

}

**Absolute loader**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void main(){

FILE \*objectcode;

int i,j,length;

char line[50],arr[20];

objectcode=fopen("objectcode.txt","r");

fscanf(objectcode,"%s",line);

printf("The program name is:");

for(int i=2;i<4;i++){

printf("%c",line[i]);

}

printf("\n");

do{

fscanf(objectcode,"%s",line);

if(line[0]=='T'){

for(i=0;i<6;i++){

arr[i]=line[i+2];

}

arr[i]='\0';

length=atoi(arr);

i=12;

while(line[i]!= '\0'){

if(line[i]!= '^'){

printf("%d:\t %c%c\n",length,line[i],line[i+1]);

length++;

i=i+2;

}

else{

i++;

}

}

}else if(line[0]=='E'){

break;

}

}

while(!feof(objectcode));

fclose(objectcode);

}

**Bankers**

#include<stdio.h>

#include<stdlib.h>

void main(){

int p\_no,r\_no,count=0,flag=0,k=0;

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

scanf("%d",&p\_no);

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

scanf("%d",&r\_no);

int allocation[p\_no][r\_no],need[p\_no][r\_no],max[p\_no][r\_no],available[r\_no],safeseq[p\_no],visited[p\_no];

printf("enter the allocation of each process:\n");

for(int i=0;i<p\_no;i++){

for(int j=0;j<r\_no;j++){

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

}

}

printf("enter the maximum allocation of each process:\n");

for(int i=0;i<p\_no;i++){

visited[i]=0;

for(int j=0;j<r\_no;j++){

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

}

}

for(int i=0;i<p\_no;i++){

for(int j=0;j<r\_no;j++){

need[i][j]=max[i][j]-allocation[i][j];

}

}

printf("enter the available space for each resources:");

for(int i=0;i<r\_no;i++){

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

}

printf("|ALLOCATION| MAXIMUM | AVAILABLE | NEED |\n");

for(int i=0;i<p\_no;i++){

printf("|");

for(int j=0;j<r\_no;j++){

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

}

printf(" |");

for(int j=0;j<r\_no;j++){

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

}

printf(" | ");

for(int j=0;j<r\_no;j++){

printf(" %d ",available[j]);

}

printf(" | ");

for(int j=0;j<r\_no;j++){

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

}

printf("|");

printf("\n");

}

while(count<p\_no){

flag=0;

for(int i=0;i<p\_no;i++){

int executed=0;

for(int j=0;j<r\_no;j++){

if(need[i][j]<=available[j]){

executed++;

}

}

if(executed==r\_no&&visited[i]==0){

for(int j=0;j<r\_no;j++){

available[j]=available[j]+allocation[i][j];

}

flag=1;

visited[i]=1;

count++;

safeseq[k++]=i;

}

}

if(flag==0){

printf("NO SAFE SEQUENCE\n");

exit(0);

}

}

printf("Safe sequence:<");

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

printf("P%d ",safeseq[i]);

}

printf(">\n");

}

**Disk Scheduling**

#include<stdio.h>

#include<stdlib.h>

void FCFS();

void SCAN();

void CSCAN();

void main(){

int choice;

while(choice!=4){

printf("\n--->1.FCFS 2.SCAN 3.CSCAN 4.EXIT\nenter the choice:");

scanf("%d",&choice);

switch(choice){

case 1:{

FCFS();

break;

}

case 2:{

SCAN();

break;

}

case 3:{

CSCAN();

break;

}

case 4:{

printf("EXITED!!\n");

break;

}

}

}

}

void FCFS(){

int n,distance,head,seek=0;

printf("Enter the no of disk requests:");

scanf("%d",&n);

int a[20];

printf("Enter the headposition:");

scanf("%d",&head);

a[0]=head;

printf("Enter the requests:");

for(int i=1;i<=n;i++){

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

}

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

distance=abs(a[i]-a[i+1]);

printf("Head moves from %d to %d with distance=%d\n",a[i],a[i+1],distance);

seek=seek+distance;

}

printf("Total head movement=%d\n",seek);

}

void SCAN(){

int n,distance,head,seek=0,q[20],q1[20],q2[20],a=0,a1=0,a2=0,temp,x,max;

printf("Enter the no of disk requests:");

scanf("%d",&n);

printf("Enter the headposition:");

scanf("%d",&head);

printf("Enter the max range:");

scanf("%d",&max);

printf("Enter the requests:");

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

scanf("%d",&temp);

if(temp>head){

q1[a1]=temp;

a1++;

}

else{

q2[a2]=temp;

a2++;

}

}

for(int i=0;i<a1;i++){

for(int j=i+1;j<a1;j++){

if(q1[i]>q1[j]){

x=q1[i];

q1[i]=q1[j];

q1[j]=x;

}

}

}

for(int i=0;i<a2;i++){

for(int j=i+1;j<a2;j++){

if(q2[i]<q2[j]){

x=q2[i];

q2[i]=q2[j];

q2[j]=x;

}

}

}

q[a]=head;

a++;

for(int i=0;i<a1;i++){

q[a]=q1[i];

a++;

}

q[a]=max;

a++;

for(int i=0;i<a2;i++){

q[a]=q2[i];

a++;

}

q[a]=0;

for(int i=0;i<(a-1);i++){

distance=abs(q[i]-q[i+1]);

printf("Head moves from %d to %d with distance=%d\n",q[i],q[i+1],distance);

seek=seek+distance;

}

printf("Total head movement=%d\n",seek);

}

void CSCAN(){

int n,distance,head,seek=0,q[20],q1[20],q2[20],a=0,a1=0,a2=0,temp,x,max;

printf("Enter the no of disk requests:");

scanf("%d",&n);

printf("Enter the headposition:");

scanf("%d",&head);

printf("Enter the max range:");

scanf("%d",&max);

printf("Enter the requests:");

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

scanf("%d",&temp);

if(temp>head){

q1[a1]=temp;

a1++;

}

else{

q2[a2]=temp;

a2++;

}

}

for(int i=0;i<a1;i++){

for(int j=i+1;j<a1;j++){

if(q1[i]>q1[j]){

x=q1[i];

q1[i]=q1[j];

q1[j]=x;

}

}

}

for(int i=0;i<a2;i++){

for(int j=i+1;j<a2;j++){

if(q2[i]>q2[j]){

x=q2[i];

q2[i]=q2[j];

q2[j]=x;

}

}

}

q[a]=head;

a++;

for(int i=0;i<a1;i++){

q[a]=q1[i];

a++;

}

q[a]=max;

a++;

q[a]=0;

a++;

for(int i=0;i<a2;i++){

q[a]=q2[i];

a++;

}

for(int i=0;i<(a-1);i++){

distance=abs(q[i]-q[i+1]);

printf("Head moves from %d to %d with distance=%d\n",q[i],q[i+1],distance);

seek=seek+distance;

}

printf("Total head movement=%d\n",seek);

}