while(fscanf(ifp,"%s",string[++i])==1){  
 fgetc(ifp);  
}

/\* string=fgetc(ifp);  
 while(string!=EOF){  
 printf(string);  
 string=fgetc(ifp);  
 }  
 char \*args[50];  
 char \*delim=",";  
 args[0]=strtok(string, delim); //use strtok function to split string into tokens using delim  
 while(args[i]!=NULL){  
 args[++i]=strtok(NULL,delim);//add NULL to last segment  
 }  
 for(i=0;i< sizeof(args);i++){  
 printf(args[i]);  
 }  
 while(fscanf(ifp,"%s%\*[^,]%d%\*[^,]%d",taskname[i++],&arrivalt[i++],&burstt[i++])==1)  
 //obtain input from a file single character at a time  
 fgetc(ifp);\*/

//search stackflow to learn more about the fgets, strtok and strdup  
 //fgets to read the file by a string args  
 //strtok to separate the string by comma  
 //strdup to copy the string and save as a temp to avoid mistake  
/\* while(fgets(args, sizeof(args),ifp)!='\0'){  
 tok=strtok(args,",");  
 while(tok!=NULL){  
 tasknum[i]=strdup(tok);  
 printf("Task num: %s \n",tasknum[i]);  
 tok=strtok(NULL,",");  
 arrivaltime[i]=atoi(strdup(tok));  
 printf("arrival time: %d \n",arrivaltime[i]);  
 tok=strtok(NULL,",");  
 bursttime[i]=atoi(strdup(tok));  
 printf("burst time: %d \n",bursttime[i]);  
 tok=NULL;  
 i++;  
 }  
 }\*/

void RR(task task1[],scheduledtask scheduledtask1[],int taskNum,FILE \*ofp) {  
 int i = 0;  
 int completetask=0;  
 int currt = 0; // Current time  
 int total\_wt=0;  
 int currtask=0;  
 int total=0;  
 task anothertask=task1[1];  
 task currentask=task1[0];  
 for(i=0;i<taskNum;i++){  
 scheduledtask1[i].tasknum=task1[i].tasknum;  
 }  
 while(taskNum>completetask){  
 if(currentask.bt<=4&& currentask.bt>0){  
 currentask.bt=4;  
 }  
 currt+=4;  
 currentask.bt-=4;  
 task1[currtask].bt-=4;  
 if(currentask.bt==0){  
 completetask++;  
 }  
 int remain=0;  
 currtask++;  
 if(currtask>=taskNum){  
 currtask-=taskNum;  
 }  
 int i;  
 for(i=0;i<taskNum;i++) {  
 if (remain == 0) {  
 if (task1[currtask].bt == 0) {  
 currtask++;  
 if (currtask >= taskNum) {  
 currtask -= taskNum;  
 }  
 } else {  
 remain = 1;  
 }  
 }  
 }  
 anothertask=task1[currtask];  
 if(taskNum>total){  
 //if current task changed, save current task into ordered list.  
 strcpy(scheduledtask1[total].tasknum,currentask.tasknum);  
 scheduledtask1[total].et=currt;  
 scheduledtask1[total].st=currentask.at;  
 total++;  
 currentask=anothertask;  
 currentask.at=currt;  
 }else{  
 scheduledtask1 = realloc(scheduledtask1,(taskNum+1)\*sizeof(scheduledtask1));  
 strcpy(scheduledtask1[total].tasknum,currentask.tasknum);  
 scheduledtask1[total].et=currt;  
 scheduledtask1[total].st=currentask.at;  
 total++;  
 currentask=anothertask;  
 currentask.at=currt;  
 }  
 task1[completetask].wt=currt-task1[completetask].bt;  
 total\_wt+=task1[completetask].wt=currt-task1[completetask].bt;  
 }  
 float avgt=(float)total\_wt/(float)taskNum;  
 for (i=0;i<taskNum;i++) {  
 fprintf(ofp,"%s\t%d\t%d\n", scheduledtask1[i].tasknum, scheduledtask1[i].st, scheduledtask1[i].et);  
 printf("%s\t%d\t%d\n", scheduledtask1[i].tasknum, scheduledtask1[i].st, scheduledtask1[i].et);  
 }  
 fprintf(ofp,"Average Wait time is %.2f\n",avgt);  
}

/\* for(i=0;i<taskNum;i++) {  
 //strcpy(dest, src)  
 strcpy(scheduledtask1[i].tasknum,task1[i].tasknum);  
 //if at< currt, means it waits to implement  
 if(currt>task1[i].at){  
 scheduledtask1[i].st=currt;  
 }  
 else{  
 currt=task1[i].at;  
 scheduledtask1[i].st=currt;  
 }  
 //start time+burst time  
 scheduledtask1[i].et=currt+task1[i].bt;  
 //wait time start time -arrival time  
 scheduledtask1[i].wt=scheduledtask1[i].st-task1[i].at;  
 currt+=task1[i].bt;  
 if(scheduledtask1[i].wt<0){  
 scheduledtask1[i].wt=0;  
 }  
 total\_wt+=scheduledtask1[i].wt;  
 }\*/

/\*  
 task\_FCFS[0].tasknum="T1";  
 task\_FCFS[0].at=0;  
 task\_FCFS[0].bt=8;  
 task\_FCFS[1].tasknum="T2";  
 task\_FCFS[1].at=1;  
 task\_FCFS[1].bt=4;  
 task\_FCFS[2].tasknum="T3";  
 task\_FCFS[2].at=2;  
 task\_FCFS[2].bt=9;  
 task\_FCFS[3].tasknum="T4";  
 task\_FCFS[3].at=3;  
 task\_FCFS[3].bt=5;  
 task\_RR[0].tasknum="T1";  
 task\_RR[0].at=0;  
 task\_RR[0].bt=8;  
 task\_RR[1].tasknum="T2";  
 task\_RR[1].at=1;  
 task\_RR[1].bt=4;  
 task\_RR[2].tasknum="T3";  
 task\_RR[2].at=2;  
 task\_RR[2].bt=9;  
 task\_RR[3].tasknum="T4";  
 task\_RR[3].at=3;  
 task\_RR[3].bt=5;  
 task\_NPSJF[0].tasknum="T1";  
 task\_NPSJF[0].at=0;  
 task\_NPSJF[0].bt=8;  
 task\_NPSJF[1].tasknum="T2";  
 task\_NPSJF[1].at=1;  
 task\_NPSJF[1].bt=4;  
 task\_NPSJF[2].tasknum="T3";  
 task\_NPSJF[2].at=2;  
 task\_NPSJF[2].bt=9;  
 task\_NPSJF[3].tasknum="T4";  
 task\_NPSJF[3].at=3;  
 task\_NPSJF[3].bt=5;\*/

while(tok!=NULL){  
 task\_FCFS[taskNum].tasknum=strdup(tok);  
 printf("Task num: %s \n",task\_FCFS[taskNum].tasknum);  
 tok=strtok(NULL,",");  
 task\_FCFS[taskNum].at=atoi(strdup(tok));  
 printf("arrival time: %d \n",task\_FCFS[taskNum].at);  
 tok=strtok(NULL,",");  
 task\_FCFS[taskNum].bt=atoi(strdup(tok));  
 printf("burst time: %d \n",task\_FCFS[taskNum].bt);  
 taskNum++;  
 tok=NULL;  
}