23.Consider a scenario of demand paged memory. Page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. Generate a solution to find maximum acceptable page-fault rate for access time that is not more than 200 nanoseconds23.

**Code: -**

#include <stdio.h>

#include <stdlib.h>

double page\_fault\_rate();

void userInput(void);

double service\_page\_fault\_empty;

double service\_page\_fault\_modified;

double mem\_access\_time;

double times\_page\_modified;

double effective\_access\_time;

double pageFaultRate;

double service\_page\_fault\_empty\_ns;

double service\_page\_fault\_modified\_ns;

double times\_page\_modified\_per;

void main(){

int swtch;

do{

printf("Select the required option \n");

printf("1.Find the PageFault Rate\n");

printf("2.Exit");

scanf("%d",&swtch);

switch(swtch){

case 1:userInput();break;

case 2:exit(0);

}

printf("\n\n");

}while(swtch<3);

}

void userInput(){

printf("\nEnter service Page Fault [Empty|Page is not Modified][in milliseconds]");

scanf("%lf",&service\_page\_fault\_empty);

printf("Enter Service Page Fault [Modified Page][in milliseconds]");

scanf("%lf",&service\_page\_fault\_modified);

printf("Enter Memory Access Time[in nanoseconds]");

scanf("%lf",&mem\_access\_time);

printf("Enter Percentage of time the page to be replaced is modified[0-100]");

scanf("%lf",&times\_page\_modified);

printf("Enter Effective Access time[in nanoseconds]");

scanf("%lf",&effective\_access\_time);

service\_page\_fault\_empty\_ns = (service\_page\_fault\_empty\*1000000);

service\_page\_fault\_modified\_ns = (service\_page\_fault\_modified\*1000000);

times\_page\_modified\_per = (times\_page\_modified/100);

printf("\nPage Fault rate calculated For:\n");

printf("Service Page Fault[Empty|Page Not Modified]=%lf \n",service\_page\_fault\_empty\_ns);

printf("Service Page Fault [Modified Page][in nanoseconds] %lf \n",service\_page\_fault\_modified\_ns);

printf("Memory Access Time[in nanoseconds]%lf\n",mem\_access\_time);

printf("Effective Access Time %lf\n",effective\_access\_time);

pageFaultRate = page\_fault\_rate(service\_page\_fault\_empty\_ns,service\_page\_fault\_modified\_ns,mem\_access\_time,times\_page\_modified\_per,effective\_access\_time);

printf("\nMaximum Acceptable Page Fault rate = %.2e[exponential notation]",pageFaultRate);

}

double page\_fault\_rate(double servicePageFaultEmpty,double servicePageFaultMod,double memAccess,double timesPages,double effAccess){

double assume,serve;

double numErator,denOminator;

double pageFault;

assume = (1- timesPages)\*servicePageFaultEmpty;

serve = timesPages\*servicePageFaultMod;

numErator = effAccess - memAccess;

denOminator = (assume+serve);

pageFault = numErator/denOminator;

return pageFault;

}