**S.Thanuja Pavani**

**AP18110010153**

**CSE-C**

**OS LAB**

**1.Write a script to find the greatest of three numbers (numbers passed as command line parameters)**

echo "Enter Num1::"

read num1

echo "Enter Num2::"

read num2

echo "Enter Num3::"

read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]

then

echo $num1 “is the greatest Number”

elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]

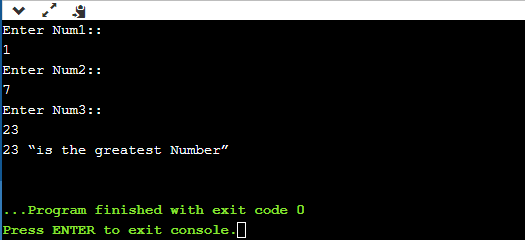
then

echo $num2 “is the greatest Number”

else

echo $num3 “is the greatest Number”

fi



**2. Write a script to check whether the given no. is even/odd**

echo "Enter a Number to find Even or Odd: "

read num

if [ $((num%2)) -eq 0 ]

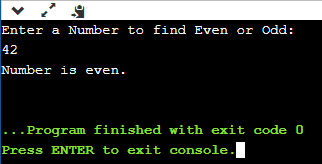
then

echo "Number is even."

else

echo "Number is odd."

fi



**3. Write a script to calculate the average of n numbers.**

echo "Enter Size:"

read N

i=1

sum=0

echo "Enter Numbers:"

while [ $i -le $N ]

do

  read num

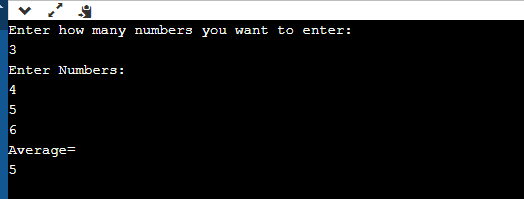
  sum=$((sum + num))

  i=$((i + 1))

done

average= expr $sum / $N

echo $average



**4. Write a script to check whether the given number is prime or not.**

echo "Enter a Number to find Prime or not: "

read number

i=2

f=0

while test $i -le `expr $number / 2`

do

if test `expr $number % $i` -eq 0

then

f=1

fi

i=`expr $i + 1`

done

if test $f -eq 1

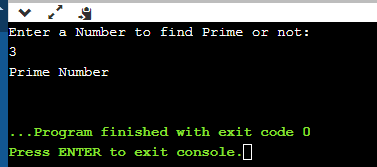
then

echo "Not a Prime Number"

else

echo "Prime Number"

fi

****

**5. Write a script to check whether the given input is a number or a string.**

read -p "Enter a number or a string: " input

if [[ $input =~ ^[+-]?[0-9]+$ ]]; then

echo "Input is an integer."

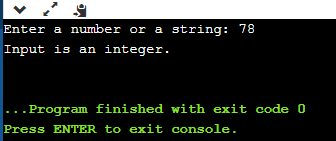
elif [[ $input =~ ^[+-]?[0-9]+\.$ ]]; then

echo "Input is a string."

else

echo "Input is a string."

Fi



**6. Write a script to print the Fibonacci series up to n terms**

echo "No.of terms:"

 read n

 a=0

 b=1

 i=2

 echo "Fibonacci Series up to $n terms :"

 echo "$a"

 echo "$b"

 while [ $i -lt $n ]

 do

 i=`expr $i + 1 `

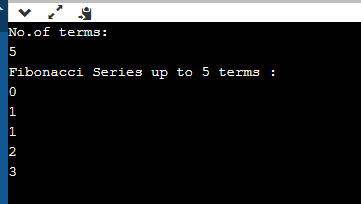
 c=`expr $a + $b `

 echo "$c"

 a=$b

 b=$c

 done



**7. Write a script to calculate the factorial of a given number**

echo "Enter a number:"

read number

factorial=1

while [ $number -gt 1 ]

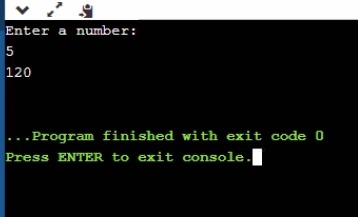
do

factorial=$((factorial \* number))

number=$((number - 1))

done

echo $factorial



**8. Write a script to calculate the sum of digits of the given number**

echo "Enter a Number:"

read n

temp=$n

s=0

sum=0

while [ $n -gt 0 ]

do

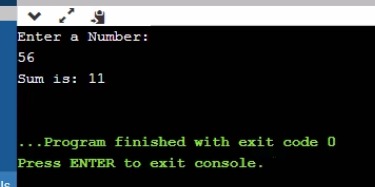
s=$(( $n % 10 ))

n=$(( $n / 10 ))

sum=$(( $sum + $s ))

done

echo "Sum is: $sum"



**9. Write a script to check whether the given string is a palindrome**

echo "Enter the string:"

read str

len=$(echo "$str" | wc -c)

while [ $len -gt 0 ]

do

ch=$(echo "$str" | cut -c $len)

s1=$s1$ch

len=`expr $len - 1`

done

if [ $s1 != $str ]

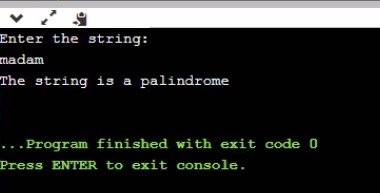
then

echo "The string is not a palindrome"

else

echo "The string is a palindrome"

fi



**10. Write a shell script that accepts a string from the terminal and echo a suitable message if it**

**doesn’t have at least 5 characters including the other symbols.**

echo "Enter the string:"

read str

length=`expr $str | wc -c`

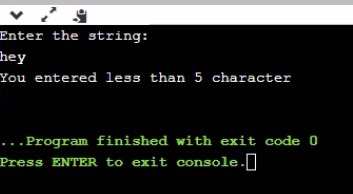
length=`expr $length - 1`

if [ $length -lt 5 ]

then

echo "You entered less than 5 character"

fi



**CPU Scheduling Algorithms**

**11. First Come First Serve**

#include <stdio.h>

int main()

{

float avgtat=0.0,avgwt=0.0;

int n;

printf("Enter number of processes::");

scanf("%d",&n);

int a[n][100];

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

for(int j=0;j<3;j++){

if(j==0){

printf("Enter process no::");

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

}

else if(j==1){

printf("Enter Arrival Time ::");

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

}

else if(j==2){

printf("Enter Burst Time ::");

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

}

}

}

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

{

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

{

if (a[i][1] > a[j][1])

{

int temp1 = a[i][0];

a[i][0] = a[j][0];

a[j][0] = temp1;

int temp2 = a[i][1];

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

a[j][1] = temp2;

int temp3 = a[i][2];

a[i][2] = a[j][2];

a[j][2] = temp3;

}

}

}

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

if(i==0){

a[i][3]=(a[i][1]+a[i][2]);

}

else{

a[i][3]=(a[i-1][3]+a[i][2]);

}

}

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

a[i][4]=(a[i][3]-a[i][1]);

avgtat+=(float)a[i][4];

}

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

a[i][5]=(a[i][4]-a[i][2]);

avgwt+=(float)a[i][5];

}

printf("P.NO AT BT CT TAT WT\n");

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

for(int j=0;j<6;j++){

printf("%d\t",a[i][j]);

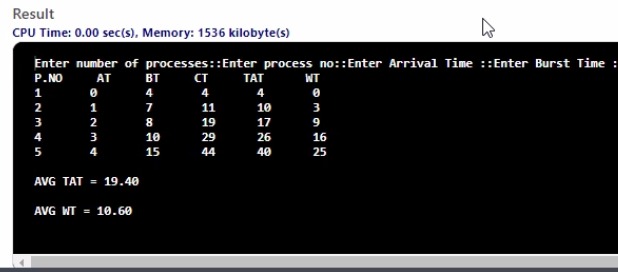
}

printf("\n");

}

printf("\nAVG TAT = %.2f\n",(avgtat/n));

printf("\nAVG WT = %.2f\n",(avgwt/n));

return 0; }

**12. Shortest Job First**

#include <stdio.h>

int main()

{

int n;

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

scanf("%d",&n);

int sjf[n][100];

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

for(int j=0;j<3;j++){

if(j==0){

printf("Enter process no::");

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

}

else if(j==1){

printf("Enter Arrival Time ::");

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

}

else if(j==2){

printf("Enter Burst Time ::");

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

}

}

sjf[i][3]=-100;

}

//Finding Arrival time 1

int k=100;

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

{

if(sjf[i][1]<k){

k=sjf[i][1];

int temp1 = sjf[0][0];

sjf[0][0] = sjf[i][0];

sjf[i][0] = temp1;

int temp2 = sjf[0][1];

sjf[0][1] = sjf[i][1];

sjf[i][1] = temp2;

int temp3 = sjf[0][2];

sjf[0][2] = sjf[i][2];

sjf[i][2] = temp3;

}

}

//Sorting

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

{

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

{

if (sjf[i][2] > sjf[j][2] )

{

int temp1 = sjf[i][0];

sjf[i][0] = sjf[j][0];

sjf[j][0] = temp1;

int temp2 = sjf[i][1];

sjf[i][1] = sjf[j][1];

sjf[j][1] = temp2;

int temp3 = sjf[i][2];

sjf[i][2] = sjf[j][2];

sjf[j][2] = temp3;

}

}

}

for (int j=1; j < (n-1); j++)

{

if (sjf[j][2]==sjf[j+1][2] )

{

if(sjf[j][1]>sjf[j+1][1]){

int temp1 = sjf[j][0];

sjf[j][0] = sjf[j+1][0];

sjf[j+1][0] = temp1;

int temp2 = sjf[j][1];

sjf[j][1] = sjf[j+1][1];

sjf[j+1][1] = temp2;

int temp3 = sjf[j][2];

sjf[j][2] = sjf[j+1][2];

sjf[j+1][2] = temp3;

}

}

}

int cnt=0,prev=0;

sjf[0][3]=(sjf[0][1]+sjf[0][2]);

while(1){

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

if(sjf[i][1]<=sjf[prev][3] && sjf[i][3]==-100){

sjf[i][3]=(sjf[prev][3]+sjf[i][2]);

prev=i;

cnt++;

break;

}

}

if(cnt==(n-1)){

break;

}

}

float avgtat=0,avgwt=0;

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

sjf[i][4]=(sjf[i][3]-sjf[i][1]);

avgtat+=(float)sjf[i][4];

}

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

sjf[i][5]=(sjf[i][4]-sjf[i][2]);

avgwt+=(float)sjf[i][5];

}

printf("P.NO AT BT CT TAT WT\n");

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

for(int j=0;j<6;j++){

printf("%d\t",sjf[i][j]);

}

printf("\n");

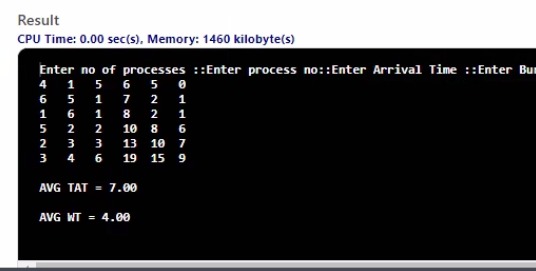
}

printf("\nAVG TAT = %.2f\n",(avgtat/n));

printf("\nAVG WT = %.2f\n",(avgwt/n));

return 0;

}



**33. Priority based scheduling**

#include <stdio.h>

int main()

{

int n;

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

scanf("%d",&n);

int priority[n][100];

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

for(int j=0;j<3;j++){

if(j==0){

printf("Enter priority no::");

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

}

else if(j==1){

printf("Enter Arrival Time ::");

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

}

else if(j==2){

printf("Enter Burst Time ::");

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

}

}

priority[i][3]=-100;

}

//Finding Arrival time 1

int k=100;

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

{

if(priority[i][1]<k){

k=priority[i][1];

int temp1 = priority[0][0];

priority[0][0] = priority[i][0];

priority[i][0] = temp1;

int temp2 = priority[0][1];

priority[0][1] = priority[i][1];

priority[i][1] = temp2;

int temp3 = priority[0][2];

priority[0][2] = priority[i][2];

priority[i][2] = temp3;

}

}

//Sorting

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

{

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

{

if (priority[i][0] > priority[j][0] )

{

int temp1 = priority[i][0];

priority[i][0] = priority[j][0];

priority[j][0] = temp1;

int temp2 = priority[i][1];

priority[i][1] = priority[j][1];

priority[j][1] = temp2;

int temp3 = priority[i][2];

priority[i][2] = priority[j][2];

priority[j][2] = temp3;

}

}

}

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

if(i==0){

priority[i][3]=(priority[i][1]+priority[i][2]);

}

else{

priority[i][3]=(priority[i-1][3]+priority[i][2]);

}

}

float avgtat=0,avgwt=0;

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

priority[i][4]=(priority[i][3]-priority[i][1]);

avgtat+=(float)priority[i][4];

}

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

priority[i][5]=(priority[i][4]-priority[i][2]);

avgwt+=(float)priority[i][5];

}

printf("\nPriority AT BT CT TAT WT\n");

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

printf("\t");

for(int j=0;j<6;j++){

printf("%d\t\t",priority[i][j]);

}

printf("\n");

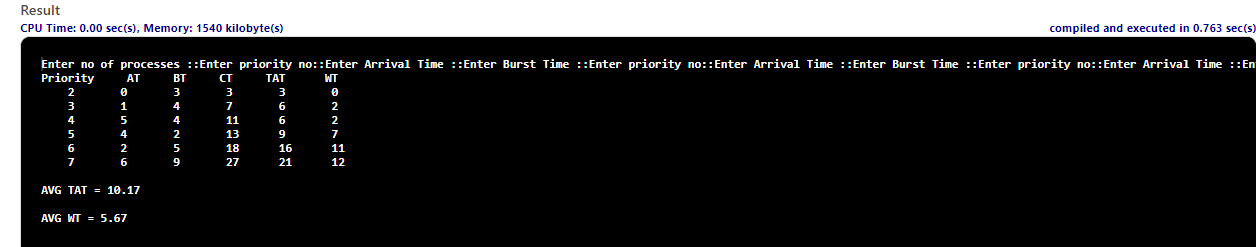
}

printf("\nAVG TAT = %.2f\n",(avgtat/n));

printf("\nAVG WT = %.2f\n",(avgwt/n));

return 0;

}



**34.Round Robin**

#include<stdio.h>

#include<math.h>

#include<string.h>

int find(int arr[],int val){

for(int i=0;i<10;i++){

if(arr[i]==val){

return 1;

}

}

return 0;

}

int main(){

int n,max=0,QuantumTime,time=0,front=0,back=0;

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

scanf("%d",&n);

printf("Enter Quantum Time ::");

scanf("%d",&QuantumTime);

int ArrivalAndBurst[1000][1000];

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

for(int j=0;j<3;j++){

if(j==0){

ArrivalAndBurst[i][j]=(i+1);

}

else if(j==1){

printf("Enter Arrival Time ::");

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

if(i==0){

time+=ArrivalAndBurst[i][j];

}

}

else if(j==2){

printf("Enter Burst Time ::");

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

max+=ArrivalAndBurst[i][j];

ArrivalAndBurst[i][7]=ArrivalAndBurst[i][j];

}

ArrivalAndBurst[i][3]=-100;

ArrivalAndBurst[i][6]=-100;

}

}

max+=time;

int ProcessQueue[1000],ind;

while(max!=time){

if(front==0 && back==0){

ind=0;

}

else{

ind=ProcessQueue[front++];

}

if( ArrivalAndBurst[ind][6]==-100){

ArrivalAndBurst[ind][6]=time;

}

if(ArrivalAndBurst[ind][2]>=QuantumTime){

ArrivalAndBurst[ind][2]-=QuantumTime;

time+=QuantumTime;

}

else if(ArrivalAndBurst[ind][2]<QuantumTime && ArrivalAndBurst[ind][2]!=0){

time+=ArrivalAndBurst[ind][2];

ArrivalAndBurst[ind][2]=0;

}

if(ArrivalAndBurst[ind][2]==0 && ArrivalAndBurst[ind][3]==-100){

ArrivalAndBurst[ind][3]=time;

}

int limit=time;

for(int p=0;p<=time;p++){

if( find(ProcessQueue,p)==0 && p!=ind && p<n){

ProcessQueue[back++]=p;

}

}

if(ArrivalAndBurst[ind][2]!=0){

ProcessQueue[back++]=ind;

}

}

float avgtat=0,avgwt=0;

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

ArrivalAndBurst[i][4]=(ArrivalAndBurst[i][3]-ArrivalAndBurst[i][1]);

avgtat+=(float) ArrivalAndBurst[i][4];

ArrivalAndBurst[i][2]=ArrivalAndBurst[i][7];

}

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

ArrivalAndBurst[i][5]=(ArrivalAndBurst[i][4]-ArrivalAndBurst[i][2]);

avgwt+=(float) ArrivalAndBurst[i][5];

}

printf("\nP.NO AT BT CT TAT WT RT\n");

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

for(int j=0;j<7;j++){

printf("%d\t ",ArrivalAndBurst[i][j]);

}

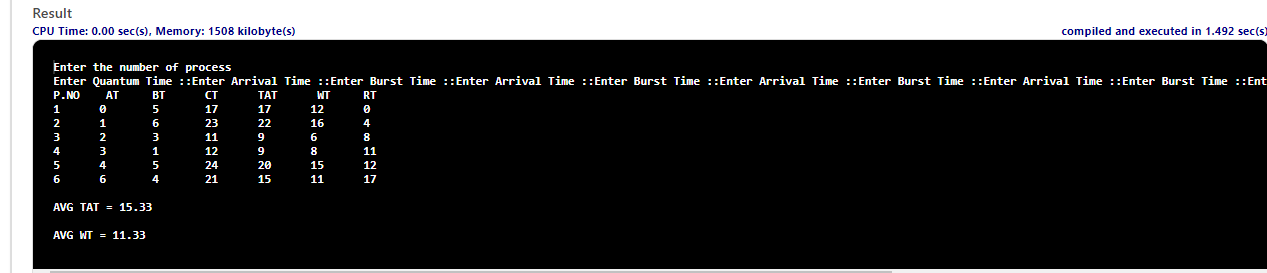
printf("\n");

}

printf("\nAVG TAT = %.2f \n",(avgtat/n));

printf("\nAVG WT = %.2f \n",(avgwt/n));

}



**35. write a C program to implement the Producer &amp; consumer Problem using Semaphore.**

#include<stdio.h>

#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;

int main()

{

int n;

void producer();

void consumer();

int wait(int);

int signal(int);

printf("\n1.Producer\n2.Consumer\n3.Exit");

while(1)

{

printf("\nEnter your choice:");

scanf("%d",&n);

switch(n)

{

case 1: if((mutex==1)&&(empty!=0))

producer();

else

printf("Buffer is full!!");

break;

case 2: if((mutex==1)&&(full!=0))

consumer();

else

printf("Buffer is empty!!");

break;

case 3:

exit(0);

break;

}

}

return 0;

}

int wait(int s)

{

return (--s);

}

int signal(int s)

{

return(++s);

}

void producer()

{

mutex=wait(mutex);

full=signal(full);

empty=wait(empty);

x++;

printf("\nProducer produces the item %d",x);

mutex=signal(mutex);

}

void consumer()

{

mutex=wait(mutex);

full=wait(full);

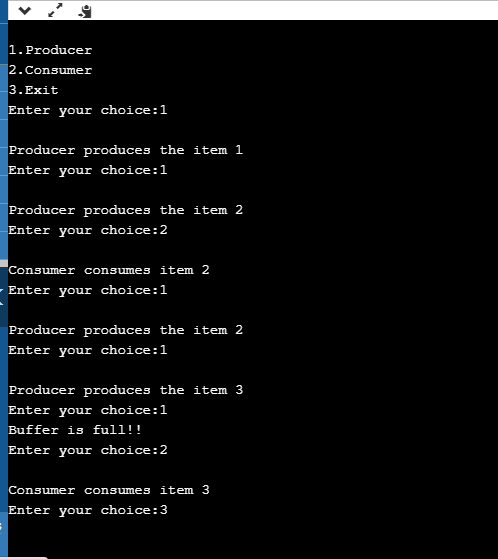
empty=signal(empty);

printf("\nConsumer consumes item %d",x);

x--;

mutex=signal(mutex);

}



**36. Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.**

#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("\n\nEnter the no of resources : ");

scanf("%d", &r);

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

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

{

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

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

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

}

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

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

{

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

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

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

}

printf("\n\nEnter 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("\n Max matrix:\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("\nProcess %d runs to completion!", 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("\nThe system is in a safe state!!\n");

printf("Safe Sequence : < ");

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

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

printf(">\n");

}

else

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

}

