2A. Compare Two Strings

```
echo enter two strings
read a
read b
if [ -z $a ]
then
echo The first string is empty
fi
if [ -z $b ]
then
echo The second string is empty
fi
if [ $a = $b ]
then
echo The strings are equal
else
echo The strings are not equal
fi
```

2B. Extract the First and Last Character from a string

```
echo enter the string
read a
first="${a:0:1}"
second="${a: -1}"
echo $first
echo $second
```

2C. Palindrome

```
echo "Enter the number"
read n
num=0
a=$n
while [ $n -gt 0 ]
do
num=$(expr $num \* 10)
k=$(expr $n % 10)
num=$(expr $num + $k)
n=$(expr $n / 10)
done
if [ $num -eq $a ]
then
echo "it is a palindrome"
else
echo "it is not a palindrome"
fi
```

3A. Factorial

```
echo enter the number of rows
read r
i=1
t=1
while [ $i -le $r ]
do
j=1
while [ $j -le $i ]
do
echo -n "*"
j=$((j+1))
done
echo
i=$((i+1))
done
```

3B. Sum of n numbers

```
echo How many numbers do you need to add
read n
i=1
sum=0
echo Enter the numbers
while [ $i -le $n ]
do
read num
sum=$((sum + num))
i=$((i + 1))
done
echo Total is $sum
```

3C. Menu Driven Program to Perform Arithmetic Operation

```
echo Enter two numbers
read a
read b
echo MENU
echo 1.Addition 2.Subraction 3.Multiplication 4.Division
echo Select from the above menu
read c
case $c in
1) echo sum = $(expr $a + $b);;
2) echo Difference = $(expr $a - $b);;
3) echo Product = $(expr $a \* $b);;
4) echo Quotient = $(expr $a / $b);;
5) echo Invalid Choice
esac
```

3D. Print * Pattern

```
echo enter the number of rows
read r
 i=1
 t=1
 while [ $i -le $r ]
 do
 j=1
 while [ $j -le $i ]
 do
 echo -n "*"
 j=$((j+1))
 done
 echo
 i=$((i+1))
 done
```

4A. Convert the characters of file from Lowercase to Uppercase

```
echo "Enter the filename"
read name
if [ ! -f $name ]
then
echo filename $name does not exist
exit 1
fi
tr "[a-z]" "[A-Z]" < $name
```

4B. Count the Number of characters, words and lines in a given text file.

```
echo Enter the filename
read file
c=`cat $file | wc -c`
w=`cat $file | wc -w`
l=`grep -c "." $file`
echo Number of characters in $file is $c
echo Number of Words in $file is $w
echo Number of lines in $file is $l
```

4C. Check if the given file exists, if not create a new file.

```
echo enter the file name
read file
if [ -f $file ]
then
echo File exists
else
echo File does not exist
touch $file
echo $file has been created
fi
```

5. Display Various System Information

```
echo “Hello ,$LOGNAME”
echo “Current Date is = $(date)”
echo “User is ‘who I am’”
echo “Current Directory = $(pwd)”
echo "Network Name and Node Name = $(uname -n)"
echo "Kernal Name =$(uname -s)"
echo "Kernal Version=$(uname -v)"
echo "Kernal Release =$(uname -r)"
echo "Kernal OS =$(uname -o)"
echo “Proessor Type = $(uname -p)”
echo “Kernel Machine Information = $(uname –m)”
echo "All Information =$(uname -a)"
```

6. Write a shell script to Manipulate Date/Time/Calendar.

```
echo "Date in various forms"
echo $(date)
echo "Today is $(date +'%m/%d/%y')"
echo "Today is $(date +'%Y-%m-%d')"
echo "Calender is various form"
echo $(cal 9 2024)
echo $(cal 2024)
echo $(cal -m May)
echo "Time in various formats"
echo $(date +"%T")
echo $(date +"%r")
echo $(date +"%I:%S:%M")
```

7A. First Come First Serve

```
#include <stdio.h>
int main()
{
   int pid[15];
   int bt[15];
   int n;
   printf("Enter the number of processes: ");
   scanf("%d",&n);
   printf("Enter process id of all the processes: ");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&pid[i]);
   }
   printf("Enter burst time of all the processes: ");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&bt[i]);
   }
   int i, wt[n];
   wt[0]=0;
   for(i=1; i<n; i++)
   {
      wt[i]= bt[i-1]+ wt[i-1];
   }
   printf("Process ID     Burst Time     Waiting Time     TurnAround Time\n");
   float twt=0.0;
   float tat= 0.0;
   for(i=0; i<n; i++)
   {
      printf("%d\t\t", pid[i]);
      printf("%d\t\t", bt[i]);
      printf("%d\t\t", wt[i]);
      printf("%d\t\t", bt[i]+wt[i]);
      printf("\n");
      twt += wt[i];
      tat += (wt[i]+bt[i]);
   }
   float att,awt;
   awt = twt/n;
   att = tat/n;
   printf("Avg. waiting time= %f\n",awt);
   printf("Avg. turnaround time= %f",att);
}
```

7B. Shortest Job First

```
#include <stdio.h>
int main()
{
    int A[100][4];
    int i, j, n, total = 0, index, temp;
    float avg_wt, avg_tat;
    printf("Enter number of process: ");
    scanf("%d", &n);
    printf("Enter Burst Time:\n");
    for (i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &A[i][1]);
        A[i][0] = i + 1;
    }
    for (i = 0; i < n; i++) {
        index = i;
        for (j = i + 1; j < n; j++)
            if (A[j][1] < A[index][1])
                index = j;
        temp = A[i][1];
        A[i][1] = A[index][1];
        A[index][1] = temp;

        temp = A[i][0];
        A[i][0] = A[index][0];
        A[index][0] = temp;
    }
    A[0][2] = 0;
    for (i = 1; i < n; i++) {
        A[i][2] = 0;
        for (j = 0; j < i; j++)
            A[i][2] += A[j][1];
        total += A[i][2];}
    avg_wt = (float)total / n;
    total = 0;
    printf("P     BT     WT     TAT\n");
    for (i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
        printf("P%d     %d     %d      %d\n", A[i][0],
            A[i][1], A[i][2], A[i][3]);}
    avg_tat = (float)total / n;
    printf("Average Waiting Time= %f", avg_wt);
    printf("\nAverage Turnaround Time= %f", avg_tat);
}
```

7C. Priority Scheduling

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int nop,t,wt[10],twt,tat[10],ttat,i,j,p[10],b[10],tmp;
 float awt, atat;
 clrscr();
 awt=0.0;
 atat=0.0;0
 printf("Enter the number of process:");
 scanf("%d",&nop);
for(i=0;i<nop;i++)
{
 printf("Enter the burst time of Process %d:",i);
 scanf("%d",&b[i]);
}
for(i=0;i<nop;i++)
 printf("Enter the priority number of each Process %d:",i);
 scanf("%d",&p[i]);
}
 for(i=0;i<nop;i++)
 {
  for(j=i+1;j<nop;j++)
  {
   if(p[i]>p[j])
   {
    t=p[i];
    p[i]=p[j];
    p[j]=tmp
```

```
tmp=b[i];
b[i]=b[j];
b[j]=tmp;
    }
   }
  }
 wt[0]=0;
  tat[0]=b[0];
  twt=wt[0];
  ttat=tat[0];
  for(i=1;i<nop;i++)
  {
   wt[i]=wt[i-1]+b[i-1];
   tat[i]=wt[i]+b[i];
   twt+=wt[i];
   ttat+=tat[i];
  }
  awt=(float)twt/nop;
  atat=(float)ttat/nop;
  printf("Process No:\tPriority:\tBurst Time:\tWaiting Time\tTurnaround Time:\n");
  for(i=0;i<nop;i++)
   printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",i,p[i],b[i],wt[i],tat[i]);

  printf("Total TurnAround Time:%d\n",ttat);
  printf("Total Waiting Time:%d\n",twt);
  printf("Average Waiting Time:%f\n",awt);
  printf("Average Turnaround Time:%f\n",atat);
  getch();
 }
```

8. Reader - Writer Problem

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
sem_t wrt; pthread_mutex_t
mutex;
int cnt = 1; int
numreader = 0;


void *writer(void *wno)
{
   sem_wait(&wrt);     cnt = cnt*2;     printf("Writer %d
modified cnt to %d\n",(*((int *)wno)),cnt);     sem_post(&wrt);


}
void *reader(void *rno)
{
```

```
    // Reader acquire the lock before modifying numreader
pthread_mutex_lock(&mutex);     numreader++;     if(numreader == 1) {
sem_wait(&wrt); // If this id the first reader, then it will block the writer
    }
    pthread_mutex_unlock(&mutex);
    // Reading Section     printf("Reader %d: read cnt as
%d\n",*((int *)rno),cnt);


    // Reader acquire the lock before modifying numreader
pthread_mutex_lock(&mutex);     numreader--;     if(numreader == 0) {
sem_post(&wrt); // If this is the last reader, it will wake up the writer.
    }
    pthread_mutex_unlock(&mutex);
}


int main()
{


    pthread_t read[10],write[5];
pthread_mutex_init(&mutex, NULL);     sem_init(&wrt,0,1);


    int a[10] = {1,2,3,4,5,6,7,8,9,10}; //Just used for numbering the producer and consumer


    for(int i = 0; i < 10; i++) {         pthread_create(&read[i], NULL,
(void *)reader, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {         pthread_create(&write[i], NULL,
(void *)writer, (void *)&a[i]);
    }


    for(int i = 0; i < 10; i++) {
pthread_join(read[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
pthread_join(write[i], NULL);
    }


    pthread_mutex_destroy(&mutex);     sem_destroy(&wrt);


    return 0;


}
```

9. Dining Philosophers Problem

```
#include<stdio.h>
#include<conio.h>
#define LEFT (i+4) %5
#define RIGHT (i+1) %5
#define THINKING 0
#define HUNGRY 1
#define EATING 2
int state[5];
void put_forks(int);
void test(int);
void take_forks(int);
void philosopher(int i)
{
if(state[i]==0)
{
take_forks(i);
if(state[i]==EATING)
printf("\n Eating in process…");
put_forks(i);
}
}
void put_forks(int i)
{
state[i]=THINKING;
printf("\n philosopher %d completed its works",i);
test(LEFT);
test(RIGHT);
}
void take_forks(int i)
{
state[i]=HUNGRY;
test(i);
}
void test(int i)
{
if(state[i]==HUNGRY && state[LEFT]!=EATING && state[RIGHT]!=EATING)
{
printf("\n philosopher %d can eat",i);
state[i]=EATING;
}
}
void main()
{
int i;
clrscr();
for(i=1;i<=5;i++)
state[i]=0;
printf("\n\t\t\t Dining Philosopher Problem");
```

```
printf("\n\t\t. ..... ");
for(i=1;i<=5;i++)
{
printf("\n\n the philosopher %d falls hungry\n",i);
philosopher(i);
}
getch();
}
```

10. First fit, Best Fit, Worst fit

A.first fit

```
 void firstFit(int blockSize[], int m, int processSize[], int n)
 {
     int i, j;
   {
    allocation[i] = -1;
     }
  for (i = 0; i < n; i++)
   {
   for (j = 0; j < m; j++)
    {
    if (blockSize[j] >= processSize[i])
    {
   blockSize[j] -= processSize[i];
   break;}}}
   printf("\nProcess No.\tProcess Size\tBlock no.\n");
 for (int i = 0; i < n; i++)
 {
 printf(" %i\t\t\t", i+1);
 printf("%i\t\t\t\t", processSize[i]);
  if (allocation[i] != -1)
  printf("%i", allocation[i] + 1);
 else
  printf("Not Allocated");
 printf("\n");
     }  }
int m;
 firstFit(blockSize, m, processSize, n);
return 0 ;
}
```

B.best fit:

```
 blockSize[], int m, int processSize[], int n)
 {{{{           i
 if (blockSize[j] >= processSize[i])
 {
 if (bestIdx == -1)
 bestIdx = j;
 else if (blockSize[bestIdx] > blockSize[j])
 bestIdx = j; }}
 {
  blockSize[bestIdx] -= processSize[i];
   }
   }
```

```
  printf( "\nProcess No.\tProcess Size\tBlock no.\n");
 for (int i = 0; i < n; i++)
 {
 printf( "%d \t\t%d\t\t",i+1,processSize[i]);
 if (allocation[i] != -1)
  printf("%d",allocation[i] + 1);
 else
 printf("Not Allocated");
 printf("\n");   }}{
 int blockSize[] = {100, 500, 200, 300, 600};
  int processSize[] = {212, 417, 112, 426};
  int m = sizeof(blockSize) / sizeof(blockSize[0]);
  int n = sizeof(processSize) / sizeof(processSize[0]);
 bestFit(blockSize, m, processSize, n);
return 0;}
```

C.Worst fit:

```
 #include<stdio.h>
 int n
 {{{
 if (blockSize[j] >= processSize[i])
 {
 if (wstIdx == -1)
 wstIdx = j;
 else if (blockSize[wstIdx] < blockSize[j])
  wstIdx = j;  }}
  printf("\nProcess No.\tProcess Size\tBlock no.\n");
  for (int i = 0; i < n; i++)
 {
  printf( "%d \t\t%d\t\t",i+1,processSize[i]);
  if (allocation[i] != -1)
 printf("%d",allocation[i] + 1);
 else
 printf("Not Allocated");
 printf("\n");   }}

 {
    int blockSize[] = {100, 500, 200, 300, 600};
  int processSize[] = {212, 417, 112, 426};
  int m = sizeof(blockSize)/sizeof(blockSize[0]);
 int n = sizeof(processSize)/sizeof(processSize[0]);
worstFit(blockSize, m, processSize, n);
return 0 ;
}
```

11. Bankers Algorithm

```
#include <stdio.h>
int main()
{
    int n, m, i, j, k;
    n = 5; // Number of processes
    m = 3; // Number of resources
    int alloc[5][3] = { { 0, 1, 0 }, // P0
                        { 2, 0, 0 }, // P1
                        { 3, 0, 2 }, // P2
                        { 2, 1, 1 }, // P3
                        { 0, 0, 2 } }; // P4
  int max[5][3] = { { 7, 5, 3 }, // P0
                    { 3, 2, 2 }, // P1
                    { 9, 0, 2 }, // P2
                    { 2, 2, 2 }, // P3
                    { 4, 3, 3 } }; // P4

    int avail[3] = { 3, 3, 2 };
   int f[n], ans[n], ind = 0;
    for (k = 0; k < n; k++) {
        f[k] = 0;
    }
    int need[n][m];
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    }
    int y = 0;
    for (k = 0; k < 5; k++) {
        for (i = 0; i < n; i++) {
            if (f[i] == 0) {

                int flag = 0;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > avail[j]){
                        flag = 1;
                         break;
                    }
                }

                if (flag == 0) {
                    ans[ind++] = i;
                    for (y = 0; y < m; y++)
                        avail[y] += alloc[i][y];
                    f[i] = 1;}}}}
```

```
    int flag = 1;

    for(int i=0;i<n;i++)
   {
    if(f[i]==0)
    {
      flag=0;
       printf("The following system is not safe");
      break;
    }
   }

    if(flag==1)
   {
    printf("Following is the SAFE Sequence\n");
    for (i = 0; i < n - 1; i++)
      printf(" P%d ->", ans[i]);
    printf(" P%d", ans[n - 1]);
   }
   return (0);
}
```

12. Producer Consumer Problem

```
#include<stdio.h>
#include<conio.h>
int main()
{
int s,n,b=0,p=0,c=0;
clrscr();
printf("\n producer and consumer problem");
do
{
printf("\n menu");
printf("\n 1.producer an item");
printf("\n 2.consumer an item");
printf("\n 3.add item to the buffer");
printf("\n 4.display status");
printf("\n 5.exit");
printf("\n enter the choice");
scanf("%d",&s);
switch(s)

{
case 1:
```

```
 p=p+1;
 printf("\n item to be produced");
 break;
 case 2:
 if(b!=0)
 {
 c=c+1;
 b=b-1;
 printf("\n item to be consumed");
 }
 else
 {
 printf("\n the buffer is empty please wait...");
 }
 break;
 case 3:
 if(b<n)
 {
 if(p!=0)
 {
 b=b+1;
 printf("\n item added to buffer");
 }
 else
 printf("\n no.of items to add...");
 }
 else
 printf("\n buffer is full,please wait");
 break;
 case 4:
 printf("no.of items produced :%d",p);
 printf("\n no.of consumed items:%d",c);
 printf("\n no.of buffered item:%d",b);
 break;
case 5:exit(0);}}
 while(s<=5);
 getch();
 return 0;
 }
```

13.Page Replacement

```
#include < stdio.h >
int main()
{
   int incomingStream[] = {4 , 1 , 2 , 4 , 5};
   int pageFaults = 0;
   int frames = 3;
   int m, n, s, pages;
   pages = sizeof(incomingStream)/sizeof(incomingStream[0]);
   printf(" Incoming \ t Frame 1 \ t Frame 2 \ t Frame 3 ");
```

```
    int temp[ frames ];
    for(m = 0; m < frames; m++)
    {
       temp[m] = -1;
    }
    for(m = 0; m < pages; m++)
    {
       s = 0;
       for(n = 0; n < frames; n++)
       {
          if(incomingStream[m] == temp[n])
          {
             s++;
             pageFaults--;  }}
       pageFaults++;
       if((pageFaults <= frames) && (s == 0))
       {  temp[m] = incomingStream[m];
       }
       else if(s == 0)
       {temp[(pageFaults - 1) % frames] = incomingStream[m];
       }
       printf("\n");
       printf("%d\t\t\t",incomingStream[m]);
       for(n = 0; n < frames; n++)
       {
          if(temp[n] != -1)
             printf(" %d\t\t\t", temp[n]);
          else
             printf(" - \t\t\t");  }}
    printf("\nTotal Page Faults:\t%d\n", pageFaults);
    return 0;
}
```