OS&CD LAB

5. A) Simulate the Multiprogramming with a fixed number of tasks (MFT).

B) Write C program to compute the First Sets for the given Grammar.

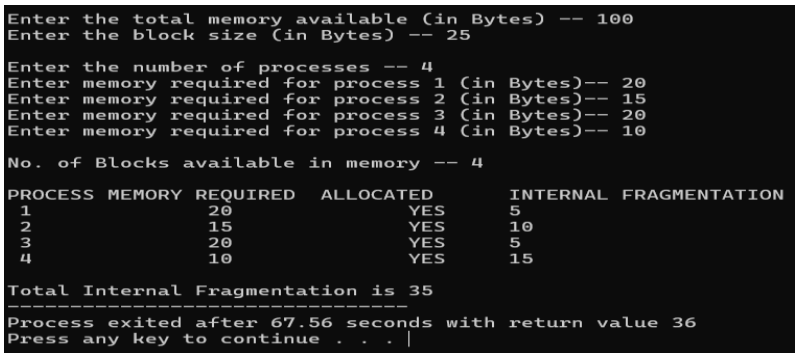
A.Simulate the Multiprogramming with a fixed number of tasks (MFT).

CODE:

```
#include<stdio.h>
main()
{
int ms, bs, nob, ef,n, mp[10],tif=0;
int i,p=0;
printf("Enter the total memory available (in Bytes) -- ");
scanf("%d",&ms);
printf("Enter the block size (in Bytes) -- ");
scanf("%d", &bs);
nob=ms/bs;
ef= ms-nob*bs;
printf("\nEnter the number of processes -- ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter memory required for process %d (in Bytes)-- ",i+1);
```

```
scanf("%d",&mp[i]);
}
printf("\nNo. of Blocks available in memory -- %d",nob);
printf("\n\nPROCESS\tMEMORY REQUIRED\t ALLOCATED\tINTERNAL FRAGMENTATION");
for(i=0;i<n && p<nob;i++)
{
printf("\n %d\t\t%d",i+1,mp[i]);
if(mp[i] > bs)
printf("\t\tNO\t\t---");
else
{
printf("\t\tYES\t%d",bs-mp[i]);
tif = tif + bs-mp[i];
p++;
getch();
}
}
if(i<n)
printf("\nMemory is Full, Remaining Processes cannot be accomodated");
printf("\n\nTotal Internal Fragmentation is %d",tif+ef);
}
```

OUTPUT:



B) Write C program to compute the First Sets for the given Grammar.

CODE:

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

int n = 0; // Counter for the First set
char first[10]; // Array to store the First set
char production[10][10]; // Array to store productions
int count; // Number of productions

void findfirst(char c, int q1, int q2) {
    int j;

    // The case where we encounter a Terminal
    if (!isupper(c)) {
        first[n++] = c;
        return;
    }

    for (j = 0; j < count; j++) {
        if (production[j][0] == c) {
            if (production[j][2] == '#') { // Epsilon production
                if (production[q1][q2] == '\0')
                    first[n++] = '#';
```

```
            else if (production[q1][q2] != '\0' && (q1 != 0 || q2 != 0)) {
                findfirst(production[q1][q2], q1, q2 + 1);
            } else {
                first[n++] = '#';
            }
        } else if (!isupper(production[j][2])) { // Terminal
            first[n++] = production[j][2];
        } else { // Non-Terminal
            findfirst(production[j][2], j, 3);
        }
    }
  }
}

void calculateFirst() {
    char c;
    int i,k;

    printf("Enter the number of productions: ");
    scanf("%d", &count);
    printf("Enter the productions (e.g., S=AB, A=a, B=b):\n");
    for (i = 0; i < count; i++) {
        scanf("%s", production[i]);
    }

    for (i = 0; i < count; i++) {
        c = production[i][0];
```

```
        n = 0; // Reset First set index
        printf("First(%c) = { ", c);
        findfirst(c, 0, 0);
        for (k = 0; k < n; k++) {
            printf("%c ", first[k]);
        }
        printf("}\n");
    }
}

int main() {
    calculateFirst();
    return 0;
}
```

OUTPUT:

```
Enter the number of productions: 3
Enter the productions (e.g., S=AB, A=a, B=b):
S=AB
A=a
B=b
First(S) = { a }
First(A) = { a }
First(B) = { b }

--------------------------------
Process exited after 15.42 seconds with return value 0
Press any key to continue . . .
```