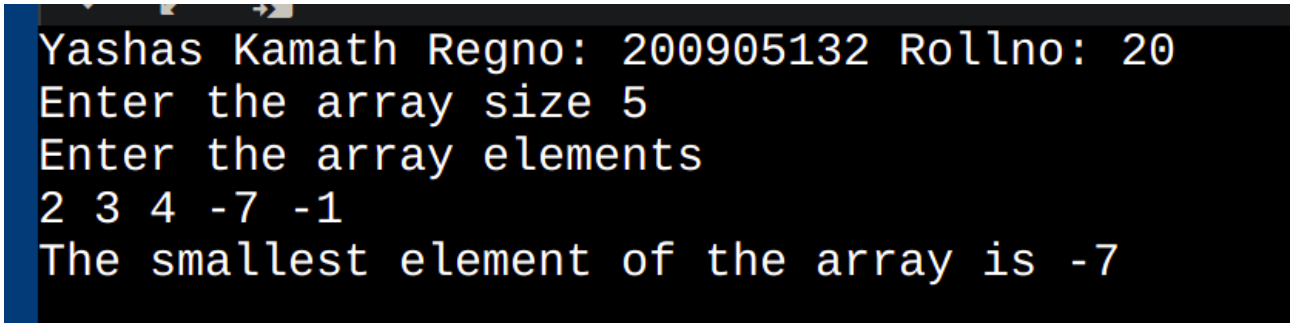
DSA lab 1

1. Write a function Smallest to find the smallest element in an array using pointer. Create a dynamically allocated array and read the values from keyboard in main. Display the result in the main function.

Solution:

```
#include<stdio.h>
#include<stdlib.h>
void Smallest(int arr[],int *ptr,int n){
        *ptr=arr[0];
        for(int i=1;i<n;i++){
                if(*(arr+i)<*ptr)
                        *ptr=*(arr+i);
        }
}
int main(){
        printf("Yashas Kamath Regno: 200905132 Rollno: 20");
        int n,*arr,result;
        printf("Enter the array size ");
        scanf("%d",&n);
        arr=(int*)calloc(n,sizeof(int));
        printf("Enter the array elements\n");
        for(int i=0;i<n;i++)
                scanf("%d",&arr[i]);
        Smallest(arr,&result,n);
        printf("The smallest element of the array is %d",result);
        return 0;
}
```

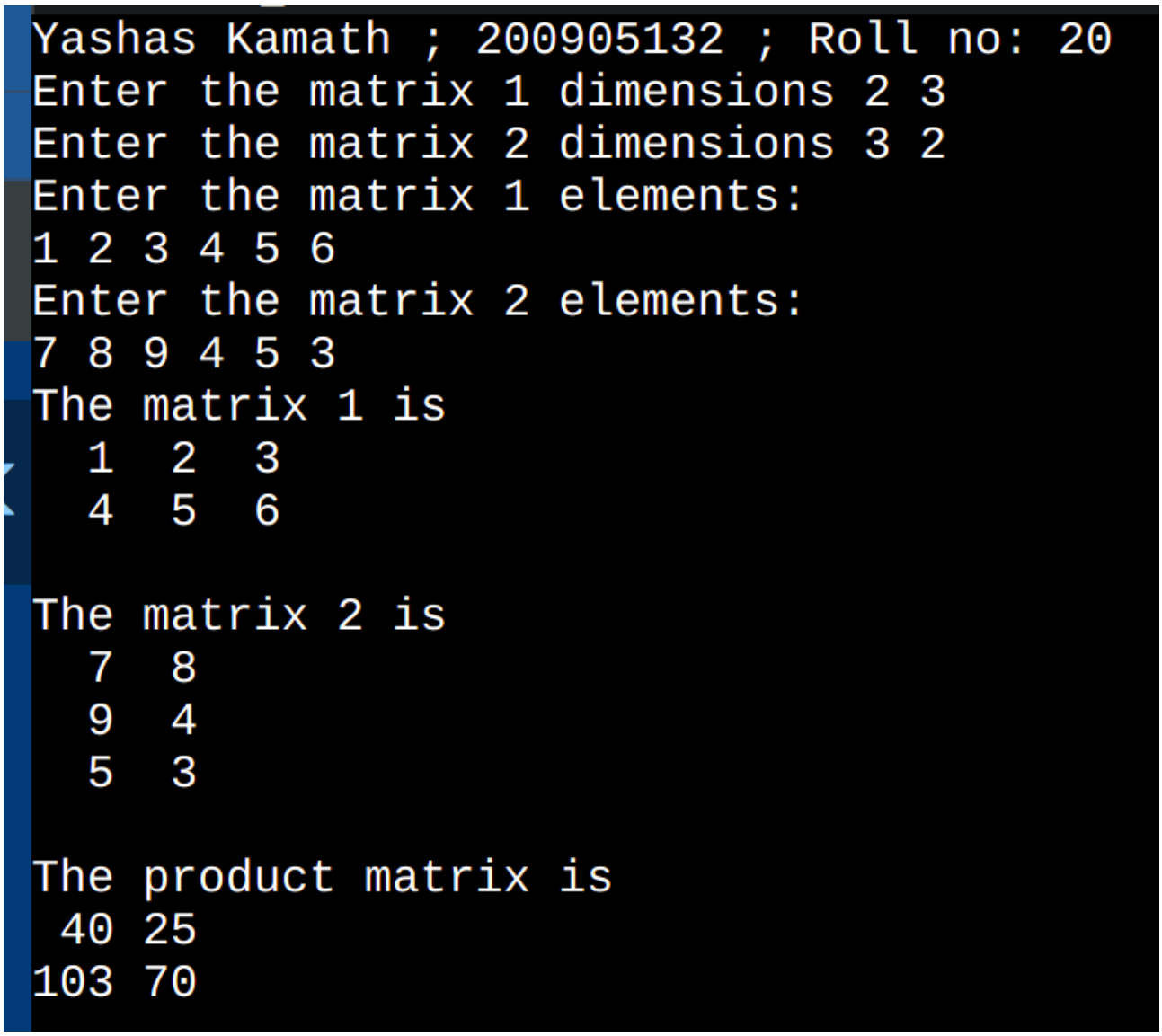


2. Implement a C program to read, display and to find the product of two matrices using functions with suitable parameters. Note that the matrices should be created using dynamic memory allocation functions and the elements are accessed using array dereferencing.

Solution:

```
#include<stdio.h>
#include<stdlib.h>
void read(int **arr,int x,int y){
        for(int i=0;i<x;i++){
           fflush(stdin);
                for(int j=0;j<y;j++)
                        scanf("%d",(*(arr+i)+j));
```

```
        }
}
void display(int **arr,int x,int y){
        for(int i=0;i<x;i++){
                for(int j=0;j<y;j++)
                        printf("%3d",*(*(arr+i)+j));
                printf("\n");
        }
        printf("\n");
}
void multiply(int **a,int **b,int **c,int m,int n,int q){
        for(int i=0;i<m;i++)
                for(int j=0;j<q;j++)
                {
                        *(*(c+i)+j)=0;
                        for(int k=0;k<n;k++)
                                *(*(c+i)+j)+=*(*(a+i)+k)*(*(*(b+k)+j));
                }
}
int main(){
   printf("Yashas Kamath ; 200905132 ; Roll no: 20\n");
        int m,n,**a,**b,**c,p,q;
        printf("Enter the matrix 1 dimensions ");
        scanf("%d %d",&m,&n);
        a=(int**)calloc(m,sizeof(int*));
        for(int i=0;i<m;i++)
                a[i]=(int*)calloc(n,sizeof(int));
        printf("Enter the matrix 2 dimensions ");
        scanf("%d %d",&p,&q);
        b=(int**)calloc(p,sizeof(int*));
        for(int i=0;i<p;i++)
                b[i]=(int*)calloc(q,sizeof(int));
        if(n!=p){
           printf("The matrices are incompatible");
           exit(0);
        }
        printf("Enter the matrix 1 elements: \n");
        read(a,m,n);
        printf("Enter the matrix 2 elements: \n");
        read(b,p,q);
        printf("The matrix 1 is \n");
        display(a,m,n);
        printf("The matrix 2 is \n");
        display(b,p,q);
        c=(int**)calloc(m,sizeof(int*));
        for(int i=0;i<m;i++)
                c[i]=(int*)calloc(q,sizeof(int));
        multiply(a,b,c,m,n,q);
        printf("The product matrix is \n");
        display(c,m,q);
        return 0; }
```



3. Samuel wants to store the data of his employees, which includes the following fields: (i) Name of the employee (ii) Date of birth which is a collection of {day, month, year} (iii) Address which is a collection of {house number, zip code and state}. Write a 'C' program to read and display the data of N employees using pointers to array of structures.

Note : You may use the following structure .

```
struct DOB {
 int day, month, year;
}
struct ADRS {
 int house_no;
 long zipcode;
 char state[20];
}
struct EMPLOYEE {
 char name[20];
 struct DOB dob;
 struct ADRS address;
```

```
}
struct EMPLOYEE emp[10];
EMPLOYEE* ptr = emp;

#include <stdio.h>
#include <stdlib.h>
typedef struct
{
int date,month,year;
}DOB;

typedef struct
{
int house_no;
long zipcode;
char state[20];
}ADRS;

typedef struct
{
char name[20];
DOB dob;
ADRS address;

}EMPLOYEE;
int main()
{
printf("Yashas Kamath Regno: 200905132 Rollno: 20");
EMPLOYEE emp[10];
EMPLOYEE *ptr = emp;
int N;
printf("Enter the number of employees : ");
scanf("%d",&N);
for(int i=0; i<N; i++)
{
printf("Enter the name of employee %d\n",i+1);
scanf("%s",(ptr+i)->name);
printf("Enter the date of birth of employee %d in date,month,year format\n",i+1);
scanf("%d%d%d",&((ptr+i)->dob.date),&((ptr+i)->dob.month),&((ptr+i)->dob.year));
printf("Enter the address of employee %d in house number, zipcode, state format\n",i+1);
scanf("%d %ld %s",&((ptr+i)->address.house_no),&((ptr+i)->address.zipcode),(ptr+i)-
>address.state);
}
printf("\n\nThe employee details are : \n");
for(int i=0; i<N; i++)
{
printf("\nThe name of employee %d is %s\n",i+1,(ptr+i)->name);
printf("The date of birth of employee %d in date-month-year format is %d-%d-%d\n",i+1,(ptr+i)-
>dob.date,(ptr+i)->dob.month,(ptr+i)->dob.year);
printf("The address of employee %d : \nHouse number - %d \nZipcode - %ld \nState - %s\n",i+1,
((ptr+i)->address.house_no),((ptr+i)->address.zipcode),((ptr+i)->address.state));
}
```

return 0;
}

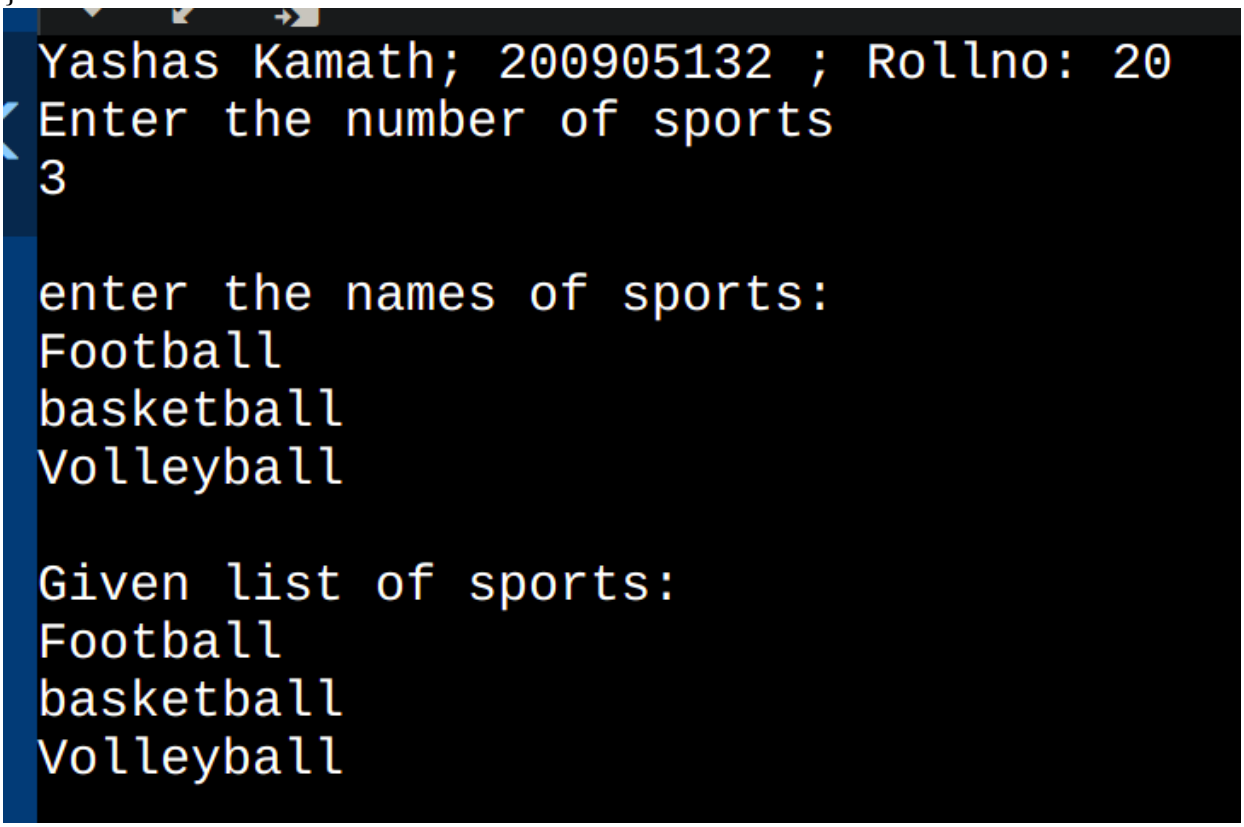
```
Yashas Kamath Regno: 200905132 Rollno: 20
Enter the number of employees : 2
Enter the name of employee 1
Ramesh
Enter the date of birth of employee 1 in date,month,year format
23 4 1965
Enter the address of employee 1 in house number, zipcode, state format
45 567 Telangana
Enter the name of employee 2
Karthik
Enter the date of birth of employee 2 in date,month,year format
9 7 2000
Enter the address of employee 2 in house number, zipcode, state format
67 890 Goa


The employee details are :

The name of employee 1 is Ramesh
The date of birth of employee 1 in date-month-year format is 23-4-1965
The address of employee 1 :
House number - 45
Zipcode - 567
State - Telangana

The name of employee 2 is Karthik
The date of birth of employee 2 in date-month-year format is 9-7-2000
The address of employee 2 :
House number - 67
Zipcode - 890
State - Goa
```

Example programs:

1) Write a program to read n names of different sports and store them using array pointers. Use dynamic memory allocation and deallocation functions. The program should display all the names and deallocate the dynamic memory at the end of the program.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(){
   printf("Yashas Kamath; 200905132 ; Rollno: 20");
 int i,n;
 char *sports[10];
 char str[100];
 printf("\nEnter the number of sports \n");
 scanf("%d", &n);
 printf("\nenter the names of sports:\n");
```

```
for (i = 0; i < n; i++)
{
scanf("%s", str);
//allocating memory equal to the length of string + 1
//Last 1 byte to accommodate the '\0'
sports[i] = (char*) calloc(strlen(str)+1, sizeof(char));
strcpy(sports[i],str);
}
printf("\nGiven list of sports: \n");
for (i = 0; i < n; i++)
printf("%s\n", sports[i]);
//Deallocate the dynamic memory
for (i = 0; i < n; i++)
free(sports[i]);
return 0;
}
```



2) Write a C program to implement a ragged array dynamically.

```
#include<stdio.h>
#include<stdlib.h>
int main(){
printf("Yashas Kamath; 200905132 ; Rollno: 20");
int rowNum, colNum, i, j;
int **table;
printf("\nenter the number of rows \n");
scanf("%d", &rowNum);
table = (int **) calloc(rowNum+1, sizeof(int *));
for (i = 0; i < rowNum; i++) /* this will tell which row we are in */
```

```
{
printf("enter size of %d row", i+1);
scanf("%d", &colNum);
table[i] = (int *) calloc(colNum+1, sizeof(int));
 printf("\n enter %d row elements \n", i+1);
 for (j = 1; j <= colNum; j++)
 { scanf("%d", &table[i][j]); }
table[i][0] = colNum;
printf("size of row number [%d] = %d\n", i+1, table[i][0]);
}
table[i] = NULL;
for (i = 0; i < rowNum; i++) /* this will tell which row we are in */
{
printf("displaying %d row elements\n", i+1);
 for (j = 0; j <= *table[i]; j++)
 {
 printf("%5d", table[i][j]);
 }
printf("\n");
}
//freeup the memory
for (i = 0; i < rowNum; i++) {
 free(table[i]);
 }
free(table);
return 0;
}
```

```
Yashas Kamath; 200905132 ; Rollno: 20
enter the number of rows
3
enter size of 1 row 2

 enter 1 row elements
1 3
size of row number [1] = 2
enter size of 2 row 2

 enter 2 row elements
3
5
size of row number [2] = 2
enter size of 3 row 3

 enter 3 row elements
4 5 3
size of row number [3] = 3
displaying 1 row elements
    2    1    3
displaying 2 row elements
    2    3    5
displaying 3 row elements
    3    4    5    3
```