Program-06

Develop a C Program for create(), read(), and display() from keyboard and to print weeks activity details report on screen.

```
# include <stdio.h>
# include <stdlib.h>

struct Day
  {
      char *dayName;
      int date;
      char *activity;
};
 struct Day *Calendar;
void create (struct Day *day)
{
    day->dayName = (char *)malloc(size of(char)*20);
    day->activity = (char *)malloc(sizeof(char)*100);
    printf ("Enter the day name:");
    scanf ("%s", day->dayName);

    printf ("enter the date:");
    scanf ("%d", &day->date);
    printf(" Enter the activity for the day");
    scanf ("%s", day -> activity);
}

void read (struct Day * calendar, int size)
{
    int i;
    for (i=0; i< size; i++)
    {
        printf(" Enter detail for day %d\n", i+1);
        create (&calendar [i]);
    }
}
void display (struct Day *calendar, int size)
{
    int i;
    printf("\ week's Activity Details: \n");
    for (i=0; i< size; i++)
    {
        printf(" Day %d \n", i+1);
        printf(" Day Name :%s\n", calendar[i].dayName);
        printf("Date %d\n", calendar[i].date);
        printf("Activity %s\n", calendar [i].activity);
    }
}
```

```
void freeMemory(struct Day *calendar, int size)
{
    int i;
    for(i=0; i<size; i++)
    {
        free(calendar[i].dayName);
        free(calendar[i].activity);
    }
}

int main()
{
    int size;
    clrscr();
    printf(" Enter the number of days in the week:");
    scanf(" %d", &size);

    calendar=(struct Day *)malloc(sizeof(struct Day)*size);

  if(calendar == NULL){
   printf("Memory allocation failed existing program\n");
    return 1;
  }

    read(calendar, size);
    display(calendar, size);
free memory(calendar, size);
 free (calendar);
 return 0;
}
```

PROGRAM - 07

Develop a menu driven program in a C for the operations on circular Queue of characters Array implementation of Que with maximum size.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX5
char circular_queue [MAX];
int front = -1; rear = -1;
int is Empty()
{
 if (front == -1 && rear == -1)
 return 1;
 else
 return 0;
}
int is Full()
{
if ((rear +1) % MAX == front)
return 1;
else
   return 0;
}
void insert Element (char element)
{
 if (is FULL())
{
 printf("Circular Queue overflow \n");
 return;
}
else if (is Empty())
{
  front  = rear=0;
}
else
{
 rear = (rear +1) % MAX;
}
 circular_queue[rear]= element
}
 void delerate Element()
 {
  if (is Empty())
 {
  if Printf("Circular Queue Underflow \n");
  return;
 }
  else if (front == rear)
  {
   front = rear= -1;
  }
 else
 {
  front = (front +1) % MAX
 }
}

 void display()
 {
  int i;
  if (is Empty())
 {
 printf("circular Queue elements:");
   i = front;
   do
  {
  printf("%c", circular_queue[i]);
    i = (i+1) % MAX;
  }
  while (i!= (rear +1) % MAX);
  printf("\n");
 }
   void main()
  {
   int choice
   char element;

   do
  {
```

```
printf ("\n\n --- Circular Queue Menu ---\n");
printf (" 1. Insert an element \n");
printf (" 2. Delete an element \n");
printf ( " 3. Display circular Queue \n ");
printf ( " 4. Exit \n ");
printf (" Enter your choice : \n ");
 scanf ( " %d ", & choice);

 switch (choice)
 {
  case1: printf (" Enter element to be inserted");
           scanf ( "%c", & element);
           insert Element (element);
           break ;
  case 2 : delete Element ();
          break ;
  case 3 : display ();
          break ;
  case 4 : Printf ("Exiting ---\n");
           break ;
  default : Printf (" In valid choice ! Please enter valid
                          option \n");
  }
 }

 while (choice != 4);
 return 0;
}
```