Program 1

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


#define NUM 7

// Structure to represent a day
typedef struct {
char *DN;    // Dynamically allocated string for the day name
int Dt;         // Date of the day
char *Act;   // Dynamically allocated string for the activity description
}DAYTYPE;

void fnFree(DAYTYPE *); void
fnDisp(DAYTYPE *); void
fnRead(DAYTYPE *);
DAYTYPE *fnCreate();

int main()
{
    // Create the calendar
    DAYTYPE *Cal = fnCreate();

    // Read data from the keyboard
fnRead(Cal);

    // Display the week's activity details
fnDisp(Cal);

    // Free allocated memory fnFree(Cal);

return 0;
}

DAYTYPE *fnCreate()
{
    DAYTYPE *c = (DAYTYPE *)malloc(NUM * sizeof(DAYTYPE));


for(int i = 0; i< NUM; i++)
               {
c[i].DN = NULL; c[i].Dt
= 0;
c[i].Act = NULL;
    }

    return c;
}

void fnRead(DAYTYPE *c)
{
```

```
              char Ch;
   for(int i = 0; i< NUM; i++)
              {
printf("\nDo you want to enter details for day %d [Y/N]: ", i + 1); scanf("%c",
&Ch); getchar();

       if(tolower(Ch) == 'n')
              continue;

printf("Day Name: ");
char nameBuffer[50];
scanf("%s", nameBuffer);
       c[i].DN = strdup(nameBuffer);  // Dynamically allocate and copy the string


printf("Date: "); scanf("%d",
&c[i].Dt);


printf("Activity: ");         char
activityBuffer[100];
scanf(" %[^\n]", activityBuffer);  // Read the entire line, including spaces
c[i].Act = strdup(activityBuffer);

printf("n");
getchar();                          //remove trailing enter character in input buffer
    }
}

void fnDisp(DAYTYPE *c)
{ printf("\nWeek's Activity
Details:\n");
   for(int i = 0; i< NUM; i++)
              {
printf("Day %d:\n", i + 1);
                          if(c[i].Dt == 0)
                          {
                                      printf("No Activity\n\n");
                                      continue;
                          }

printf("  Day Name: %s\n", c[i].DN); printf("
Date: %d\n", c[i].Dt);
printf("  Activity: %s\n", c[i].Act);
    }
}

void fnFree(DAYTYPE *c)
{
   for(int i = 0; i< NUM; i++)
              {
       free(c[i].DN);
free(c[i].Act);
```

```
    }
free(c);
}
```

Program 2

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

int main() {     char st[200], srch[30], rep[30],
res[200], cpy[200];     int i=0, j=0 ,k=0, l, mtch,
iStop, len, nom=0;

    printf("\nEnter the main string\n");
              scanf(" %[^\n]", st);

    printf("\nEnter the Pattern string\n");
              scanf(" %[^\n]", srch);

    printf("\nEnter the Replace string\n");
              scanf(" %[^\n]", rep);

              strcpy(cpy, st);

    for(i=0;i<(strlen(st)-strlen(srch)+1);i++)
    {
        mtch = 0;

        for(j=0;j<strlen(srch);j++)
        {

            if(st[i+j] == srch[j])
            {
                mtch++;
            }
else            {
break;
            }

            if(mtch == strlen(srch))   //Check if number of character matches equals length of pattern string
            {
                nom++;      //update number of total matches by 1
for(k=0;k<i;k++)
                {
                    res[k] = st[k];     //copy till the ith character where the match occured
                }
                iStop = k + strlen(srch); //point from where rest of the original string has to be copied
res[k] = '\0';
                strcat(res, rep); // append the replacement string
len = strlen(res);

                for(k=iStop, l=0; st[k] != '\0';k++, l++) //copy rest of original string
                {
                    res[len+l] = st[k];
                }
                res[len+l] = '\0';
strcpy(st,res);
            }
        }
    }

    printf("\nInput Text\n");
printf("%s\n",cpy);

    if(nom > 0)
    {
        printf("\n%d matches occured\n\nText after replacing matched patterns is shown below\n", nom);
printf("\n%s\n",res);
    }
else
    {
        printf("\nPattern String not found in Text\n");
    }
return 0;
}
```

Program 3

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 4

bool isFull(int top) {
    return top == MAX - 1;
}

bool isEmpty(int top) {
    return top == -1;
}

void push(int stk[], int elem, int *top) {
    if (!isFull(*top)) {
        stk[++(*top)] = elem;
    }
}

int pop(int stk[], int *top) {
    return isEmpty(*top) ? -1 : stk[(*top)--];
}

void display(int stk[], int top) {
    if (isEmpty(top)) {
        printf("\nStack Empty\n");
        return;
    }
    for (int i = top; i >= 0; i--) {
        printf("\t%d\n", stk[i]);
    }
    printf("Stack has %d elements\n", top + 1);
}

int peek(int stk[], int top) {
    return isEmpty(top) ? -1 : stk[top];
}

bool isPalindrome(int num) {
    int rev = 0, original = num;
    while (num) {
        rev = rev * 10 + num % 10;
        num /= 10;
    }
    return rev == original;
}

int main(void) {
    int stk[MAX], top = -1, elem, choice;

    while (1) {
        printf("\n1. Push\n2. Pop\n3. Display\n4. Peek\n5. Check Palindrome\n6. Exit\nChoice: ");
        scanf("%d", &choice);  // Fixed: Removed extra &ch

        switch (choice) {
            case 1:
                if (isFull(top)) {
                    printf("\nStack Overflow\n");
                } else {
                    printf("\nEnter element: ");
                    scanf("%d", &elem);
                    push(stk, elem, &top);
                }
                break;
            case 2:
                if (isEmpty(top)) {
                    printf("\nStack Underflow\n");
                } else {
                    elem = pop(stk, &top);
                    printf("\nPopped Element: %d\n", elem);
                }
                break;
            case 3:
                display(stk, top);
                break;
            case 4:
                printf("\nTop Element: %d\n", peek(stk, top));
                break;
            case 5:
                printf("\nEnter number: ");
                scanf("%d", &elem);
                if (isPalindrome(elem))  // Fixed: was ispalindrome(), corrected to isPalindrome()
                    printf("\n%d is a palindrome\n", elem);
                else
                    printf("\n%d is not a palindrome\n", elem);
                break;
            case 6:
                exit(0);
            default:
                printf("\nInvalid choice\n");
        }
    }

    return 0;
}
```