**Assignment: 7**

**Exp.1.** WAP to demonstrate the union's effectiveness over structure. You can use any previously given structure program to depict the idea.

**Coding:** #include<stdio.h>

#include<string.h>

#include<stdlib.h>

struct cricket\_player {

    char player\_name[50];

    int player\_age;

    int number\_matches;

    int avg\_run;

}c[20];

union cricket\_player\_u {

    char player\_name\_u[50];

    int player\_age\_u;

    int number\_matches\_u;

    int avg\_run\_u;

}b[20];

// for structure

int main(){

      int i,n;

      printf("Enter the number of player for structure ");

      scanf("%d",&n);

      c;

      for(i=0;i<n;i++){

      printf("Enter data of %d cricketer \n", i+1);

      printf("Player Name: ");

      scanf("\n");

      scanf("%[^\n]%\*c",c[i].player\_name);

      printf("Player Age: ");

      scanf("%d",&c[i].player\_age);

      printf("Matches Played: ");

      scanf("%d",&c[i].number\_matches);

      printf("Average runs: ");

      scanf("%d",&c[i].avg\_run);

   }

      printf("\nPlayer name \t Avg run");

      for(i=0;i<n;i++){

      printf("\n%s\t\t %d",c[i].player\_name,c[i].avg\_run);

      printf("\n");

   }

// for union

      int j,k;

      printf("Enter the number of player for union ");

      scanf("%d",&k);

      b;

      for(j=0;j<k;j++){

      printf("Enter data of %d cricketer \n", j+1);

      printf("Player Name: ");

      scanf("\n");

      scanf("%[^\n]%\*c",b[j].player\_name\_u);

      printf("Player Age: ");

      scanf("%d",&b[j].player\_age\_u);

      printf("Matches Played: ");

      scanf("%d",&b[j].number\_matches\_u);

      printf("Average runs: ");

      scanf("%d",&b[j].avg\_run\_u);

   }

      printf("\nPlayer name \t Avg run");

      for(j=0;i<k;j++){

      printf("\n%s\t\t %d",b[j].player\_name\_u,b[j].avg\_run\_u);

      printf("\n");

   }

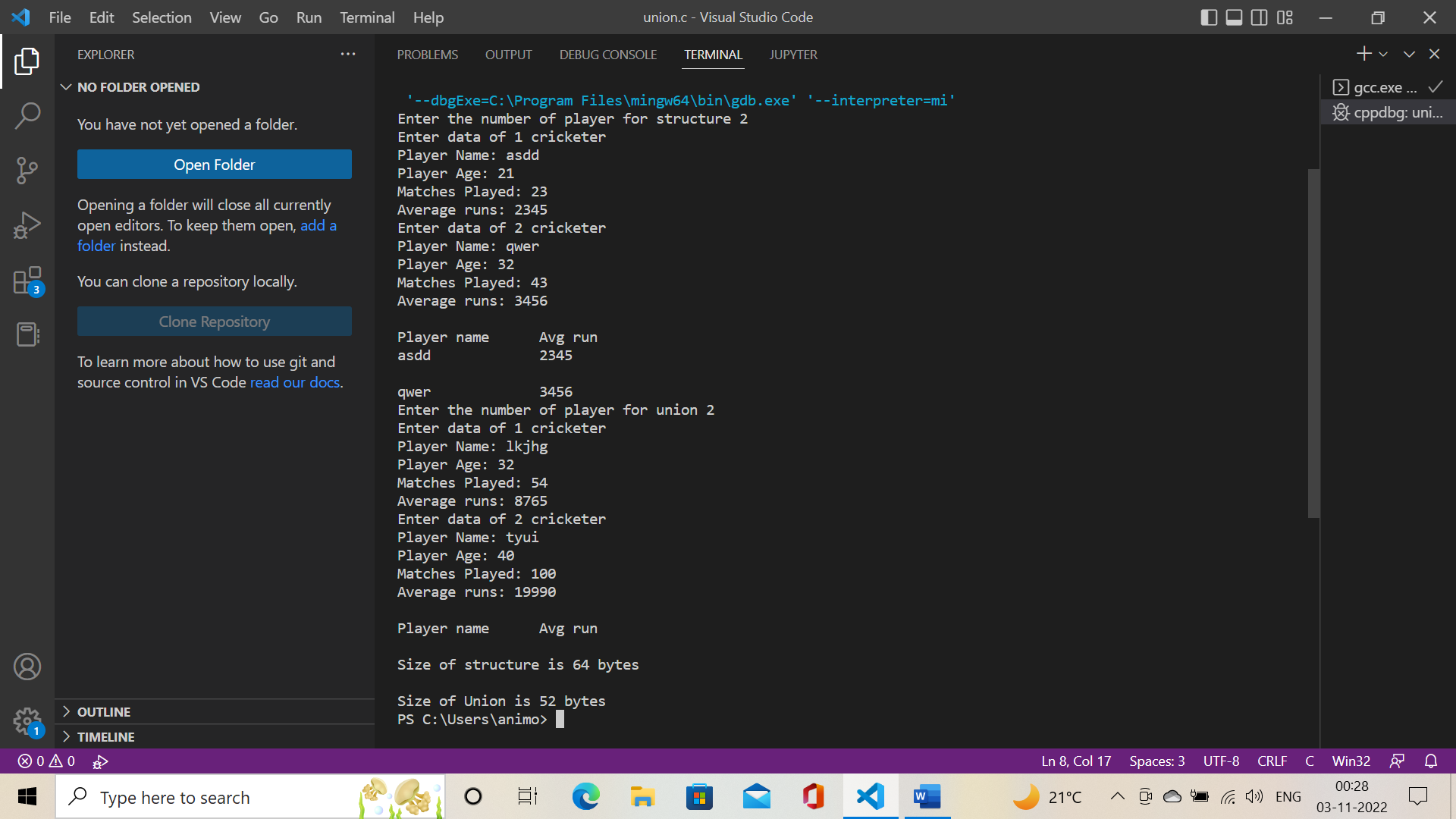
   printf("\n\nSize of structure is %lu bytes",sizeof(struct cricket\_player));

   printf("\n\nSize of Union is %lu bytes",sizeof(union cricket\_player\_u));

   return 0;

}

**Output:**



**Exp.2.** WAP to demonstrate the various run-time memory allocation approaches like

 a. Malloc

 b. Calloc

 c. Free

 d. Realloc

For implementing this, make use of array, function, and wherever necessary pointer.

**Coding:** #include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define NUMBER\_OF\_ELEMENTS 100

int main()

{

    int \*parr=calloc(NUMBER\_OF\_ELEMENTS,sizeof(int));

    if (parr==NULL)

    {

        printf("Couldn't allocate memory through calloc\n");

    }

    else

    {

        printf("Memory allocation successful with calloc\n");

    }

    int \*barr=malloc(NUMBER\_OF\_ELEMENTS\*sizeof(int));

    if(barr==NULL)

    {

        printf("Couldn't allocate memory through malloc\n");

    }

    else

    {

        memset(barr,0,NUMBER\_OF\_ELEMENTS\*sizeof(barr[0]));

        printf("Memory allocation successful with malloc\n");

    }

    int \*carr=calloc(NUMBER\_OF\_ELEMENTS,sizeof(int));

    if(carr==NULL)

    {

        printf("Couldn't allocate memory");

    }

    else

    {

        printf("Memory allocation successful\n");

        carr=realloc(carr,(NUMBER\_OF\_ELEMENTS/2)\*sizeof(int));

        if (carr==NULL)

        {

            printf("Memory reallocation fails");

        }

        else

        {

            printf("Memory reallocation successful");

        }

    }

    printf("\nsizeof parr = %d", sizeof(parr));

    free(parr);

    parr=NULL;

    return 0;

**Output:** 